DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Computer Sciences Corporation 6565 Arlington Blvd. Falls Church, Virginia 22046	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

DIN Simulator Program Reports

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Program Report

5. AUTHOR(S) *(First name, middle initial, last name)*

Richard D. Crumm

6. REPORT DATE May 1972	7a. TOTAL NO. OF PAGES 421	7b. NO. OF REFS 5

8a. CONTRACT OR GRANT NO. DCA100-71-C-0048 b. PROJECT NO.	9a. ORIGINATOR'S REPORT NUMBER(S) R413681-2-1
c. d.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. DISTRIBUTION STATEMENT

Distribution limited to U.S. Government agencies only, test and evaluation; May 1972. Other requests for this document must be referred to the Defense Communications Agency.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Defense Communications Agency Code 350 Washington, D.C. 20305

13. ABSTRACT

This report contains detailed descriptions and flow charts of the programs that constitute the DIN Simulator. It is written to assist the program analyst or programmer in program maintenance and update.

The DIN Simulator is a discrete event simulator model designed to simulate the Automatic Digital Network (AUTODIN) message store and forward network. It is a derivative of the Overseas AUTODIN Simulator written for the Philco 2000 computer in the TAC language. The DIN Simulator is written in the higher level FORTRAN and COBOL languages for execution on an IBM 360 model computer and designed for easy conversion to other systems.

The DIN Simulator not only has the capability of simulating the AUTODIN network in the same manner as the former Overseas AUTODIN Simulator but also incorporates new features which permit the simulation of almost any type of store and forward network. Thus, new concepts of operating the AUTODIN network could be modeled. The DIN Simulator also incorporates functions of the Automated AUTODIN Traffic Processing System (AATPS) which assembled network and traffic data for the Overseas AUTODIN Simulator.

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
AUTODIN Discrete Event Simulation DIN Simulator						

DIN SIMULATOR
PROGRAM REPORTS

Prepared for

DEFENSE COMMUNICATIONS AGENCY
WASHINGTON, D.C.
CONTRACT DCA100-71-C-0048

DISTRIBUTION LIMITED TO U.S. GOVERNMENT AGENCIES ONLY; TEST AND EVALUATION; MAY 1972. OTHER REQUESTS FOR THIS DOCUMENT MUST BE REFERRED TO THE DEFENSE COMMUNICATIONS AGENCY.

MAY 1972

REPORT R413681-2-1

COMPUTER SCIENCES CORPORATION

6565 Arlington Boulevard

Falls Church, Virginia 22046

Major Offices and Facilities Throughout the World

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd)

Section 6 - Cont'd

TABLE OF CONTENTS (Cont'd)

Section 11 - Cont'd

TABLE OF CONTENTS (Cont'd)

Section 16 - Cont'd

LIST OF ILLUSTRATIONS

Figure

LIST OF TABLES

Table

LIST OF TABLES (Cont'd.)

LIST OF TABLES (Cont'd.)

SECTION 1 - INTRODUCTION

1.1 GENERAL

This report contains detailed descriptions and flow charts of the programs that constitute the DIN Simulator. It is written to assist the program analyst or programmer in program maintenance and update. Detailed operating instructions for the model are contained in the DIN Simulator Users Manual, Reference 1.

The DIN Simulator is a discrete event simulator model designed to simulate the Automatic Digital Network (AUTODIN) message store and forward network. It is a derivative of the Overseas AUTODIN Simulator written for the Philco 2000 computer in the TAC language. The DIN Simulator is written in the higher level FORTRAN and COBOL languages for execution on an IBM 360 model computer and designed for easy conversion to other systems.

The DIN Simulator not only has the capability of simulating the AUTODIN network in the same manner as the former Overseas AUTODIN Simulator but also incorporates new features which permit the simulation of almost any type of store-and-forward network. Thus, new concepts of operating the AUTODIN network could be modeled. The DIN Simulator also incorporates functions of the Automated AUTODIN Traffic Processing System (AATPS) which assembled network and traffic data for the Overseas AUTODIN Simulator. Descriptions of the Overseas AUTODIN Simulator and the AATPS are given in References 2 through 5. The application of the DIN Simulator is essentially the same as that of the Overseas AUTODIN Simulator as described in Reference 5.

The DIN Simulator was developed for the IBM 360 but is generally written in standardized FORTRAN IV and American National Standards Institute (ANSI) COBOL to achieve maximum machine independence. Deviations from the ANSI COBOL were made only where the running efficiency would be improved greatly by the use of specialized COBOL instructions. Those FORTRAN programs which use structured tables and require bit manipulation cannot be entirely machine independent and still be efficient in the use of available core memory. To make the source coding machine independent, a system of pseudo-instructions and mnemonic references to structured tables was devised. A FORTRAN preprocessor program was written to interpret the pseudo-instructions and convert them to valid FORTRAN instructions for the IBM 360. To convert those programs using pseudo-instructions for use on a different system, only the preprocessor program need be rewritten to produce acceptable instructions and the mnemonic reference tables restructured as necessary. A description of the concepts and use of the Preprocessor program is contained in Section 15 and the appendices of this report.

The AUTODIN and the DIN Simulator are described briefly in this section. The remainder of the report is devoted to detailed description of the programs. This includes a utility program named DATASERV which was designed for the manipulation of card image files. While this program is not an integral part of the model it is more convenient to use for assembling input data than to rely on system utility programs which vary greatly from system to system.

1.2 SYSTEM DESCRIPTION

AUTODIN is a switched network of the Defense Communications System (DCS) composed of a series of switching centers which provide record communications to tributary subscribers in the Department of Defense and other Federal Government agencies. AUTODIN Switching Centers (ASCs) in the continental United States (CONUS) and Hawaii are the leased Automatic Electronic Switching Center (AESC) type while the overseas units are the Automatic Digital Message Switch (ADMS) type. The switching centers are interconnected by high speed transmission links which include channels in microwave, coaxial cable, submarine cable, satellite, or high frequency radio systems. The transmission rates may vary up to 4800 baud. The tributary subscribers served by AUTODIN may be as simple as an office connected by a 45-baud teletype circuit or as complex as a nonautomatic minor relay center or a computer equipped automatic distribution unit served by a 4800-baud transmission rate. Messages accepted from subscribers may be in teletype, punched card, magnetic tape, or computer format.

1.3 MODEL DESCRIPTION

1.3.1 General Description

The DIN Simulator is a highly detailed computer model which provides quantitative measures of system performance resulting from proposed changes in network configuration and routing, changes in subscriber and traffic volume, or changes in switch hardware or software design. The model is comprised of a Simulation Section and a Traffic Preprocessing Section; their data flow is illustrated in Figures 1-1 and 1-2 respectively. The Simulation Section accepts the input description of the traffic and network, performs the simulation and generates a series of reports. This part of the simulator is described in Paragraph 1.3.2. The results obtained from each simulation are summarized as a series of reports which are divided into categories such as speed of service, link utilization, in-station handling time, switch queues, link queues, and tributary queues. The speed of service report can be created by two different methods of computation. The first method measures the speed of service from the time a message begins transmission from its originating tributary until delivery is complete at the destination tributary. The second method eliminates tributary transmission times and the time spent in outgoing tributary queues and thereby measures the capabilities of the switching centers and the interswitch links. These speed of service measurements can be made on a tributary-to-tributary basis, on a switch-to-switch basis, or on an overall system basis.

The Traffic Preprocessing Section consists of a series of programs that extract traffic information from Header Extract tapes. This makes it possible to obtain input traffic based on the actual journals recorded at the ASCs. This part of the simulator is described in Paragraph 1.3.3.

1.3.2 Simulation Section

The Simulation Section is capable of independent operation using networks, routing, and traffic data entered via punched card input files. Data flow in this section is depicted in Figure 1-1. Paragraphs 1.3.2.1 through 1.3.2.7 summarize the programs which comprise this section.

1.3.2.1 Status Generator Program

Network information is supplied to the model in four input files: the Components file which describes the switching centers; the Link file which describes the inter-switch links; the Tributary file which describes the tributaries; and the Routing file which provides the routing doctrine. These files are in punched card format and, except for the Link and Tributary files, must be manually prepared. When simulating the full AUTODIN network, the Link and Tributary files generated by the Traffic Preprocessor Section may be used. There is a fifth input file, the Configuration file, which is a list of switches to be used in the simulation. If it is desired to simulate only a portion of the AUTOVON network, rather than generate a new set of Component, Link, Tributary, and Routing files which reflect only the desired portion, the desired switches are specified in the Configuration file. The program then extracts the appropriate items from the remaining files. The Status Generator program verifies the input data for correctness, and, in some cases completeness, and generates a Status Table file which reflects the network configuration. The program also generates a composite file of switch, link, and tributary inputs which cross-reference the Status Tables. This file is called the Translation file.

1.3.2.2 Statistical Message Generator

The Statistical Message Generator permits the generation of messages to be used in the simulation using statistical distributions as inputs. Message characteristics such as security, precedence, length, type, multiple address factor, and destinations are selected randomly from the statistical distributions using Monte Carlo techniques. The program output is an unsorted file of generated messages. The program uses the Translation file generated by the Status Generator for tributary name validation and transmission parameters.

1.3.2.3 Discrete Message Generator

The Discrete Message Generator generates messages to be used in simulation from the traffic data derived from AUTODIN sample data. The input to the program is the Sorted Traffic file from the Traffic Preprocessor Section and the Translation file generated by the Status Generator program. The output is an unsorted file of generated messages.

1.3.2.4 Message Sort Program

One of the model requirements is the ability to simulate outages at switches, or on links and tributaries. Another requirement is the ability to change routing doctrine during the simulation process. These and other exogenous events may be entered into the model via punched cards in the Message Sort program. Other inputs to the program are one or more message files produced by the Message Generator programs and the Translation file from the Status Generator program. The messages and optional special events are sorted into chronological order and merged into a single Event-Message file.

1.3.2.5 Basic Simulation

The inputs to the Basic Simulation program are the Status Tables file from the Status Generator program which reflects network configurations and the chronological Event-Message file from the Message Sort program. As the event-by-event simulation is performed, each major event is noted by an event description record written on either an SOS Event file or a Link Event file. The status of the network components (switches, links, and tributaries) are recorded on a Status file at an interval specified by optional inputs.

1.3.2.6 Reports Program

The simulator results are summarized by the Reports program using inputs from the SOS and Link Event files, the Status file from the Basic Simulation program, and the Translation file from the Status Generator program. The reports produced are SOS, Link Utilization, In-Station Handling Time, Switch Queue, Link Queue, and Tributary Queue.

1.3.2.7 List Events Program

In some simulation problems it may be necessary or desirable to examine the specific events or status records which occur at a particular network component, or it may be desirable to trace a particular message or series of messages throughout the simulation or to perform some analysis not provided for by the summarized reports. The List Events program provides the ability to examine these events by selecting and printing those events which fall within the confines of the specified parameters. The program inputs are the Event and Status files from the Basic Simulation program, the Translation file from the Status Generator program, and a list of parameters introduced via punched cards.

1.3.3 Traffic Preprocessing Section

The size of the AUTODIN network, in terms of numbers of tributaries and messages exchanged, makes it virtually impossible to manually prepare the input data required to effectively simulate the network. The Traffic Preprocessing Section prepares link, tributary, and message data for input to the Simulation Section using an AUTODIN data base and a traffic sample data taken monthly at AUTODIN Switching Centers. Data flow in this section is depicted in Figure 1-2. Paragraphs 1.3.3.1 through 1.3.3.6 summarize the programs which comprise this section.

1.3.3.1 Header Extract Program

The data sample taken by the AUTODIN switches becomes available to the DIN Simulator in the form of card image records. Each record reflects some message transaction made at a reporting switch. The Header Extract program validates each of the incoming records and generates two output files; one which reflects tributary transactions called the Header Extract Tributary file, and another which reflects link transactions called the Header Extract Link file.

1.3.3.2 AMIE Extract Program

A second source of information about the AUTODIN network is a data base named the AUTODIN Management Index (AMIE) file. Static link and tributary data are extracted from this file and preserved on the AMIE Link and AMIE Tributary files for use by following programs.

1.3.3.3 Link Load Program

The Link Load program uses the Header Extract Link and AMIE Link files to produce the Link file for input to the Status Generator program. This file defines links which were active during the sample period. The program also produces a listing of the actual traffic load for each link channel and a minute-by-minute channel activity report which may be used for determining outages and/or part-time channel usage.

1.3.3.4 Assemble Traffic Program

The Assemble Traffic program uses the Header Extract Tributary and AMIE Tributary files to produce the Header Extract Traffic file which reflects the end-to-end message data derived from the traffic sample. The program also generates a Tributary file for input to the Status Generator program and a listing of tributary channel usage.

1.3.3.5 Traffic Modification and Sort Program

The unsorted Header Extract Traffic file is input to the Traffic Modification and Sort program which, in addition to sorting the traffic into the logical order required by the Discrete Message Generator, provides a limited ability to modify the traffic data. It is also possible to select and/or delete traffic data so that when simulating less than the full AUTODIN network only those messages relative to the simulated portion will be generated. A listing of message activity by switch by hour is also produced so that busy hour periods at each switch may be determined and a judgement made concerning the validity of the sample data. The output file is named Sorted Traffic file.

1.3.3.6 Traffic Distribution Report

The AUTODIN Traffic Distribution Report generated by the Traffic Distribution program provides the ability to make a judgement concerning the validity of the sample and to provide a summary of the overall traffic in the network. A report is made for each network switch showing the number of messages sent by the tributaries of that switch and the distribution of those messages to other switches in the network. The report also calculates the SOS attained in the network in delivering those messages.

1.4 ORGANIZATION OF THIS REPORT

The remaining sections of this report describe the individual programs:

Program	Section
Status Generator	2
Discrete Message Generator	3
Statistical Message Generator	4
Message Sort	5
Basic Simulation	6
Reports	7
List Events	8
Header Extract	9
AMIE Extract	10
Assemble Traffic	11
Traffic Modification and Sort	12
Traffic Distribution	13
Link Load	14
FORTRAN Simulator Preprocessor	15
DATASERV	16

Also included in appendices to this report are a synopsis of the Basic Simulation programs event-by-event functions, a discussion of the various types of events used in the Basic Simulation program, a representation of the current structure of the Basic Simulation Status Tables, and a description of the concepts and use of the FORTRAN Preprocessor program. Program listings are bound under separate cover.

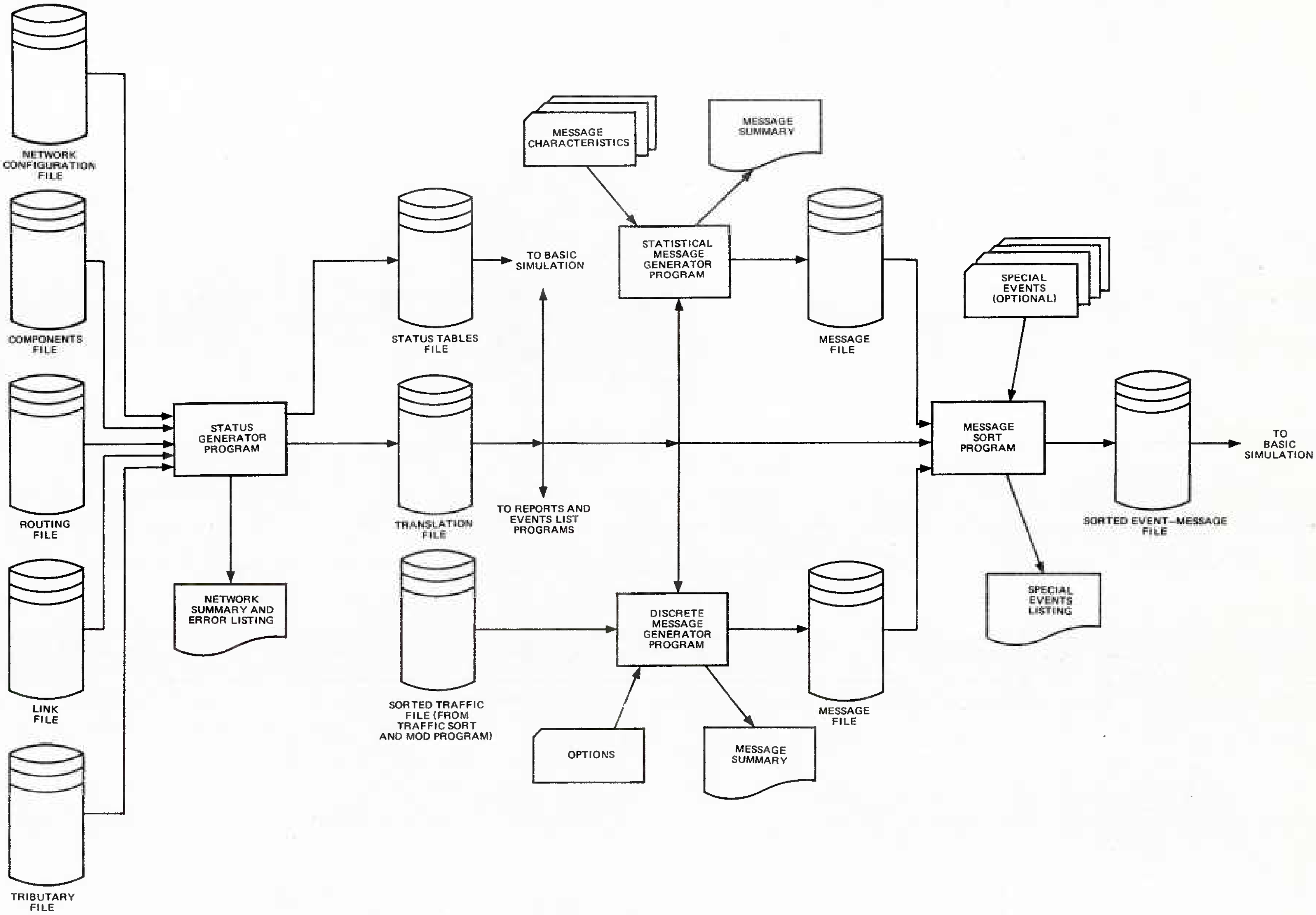


Figure 1-1. Simulation Section Data Flow (Sheet 1 of 2)

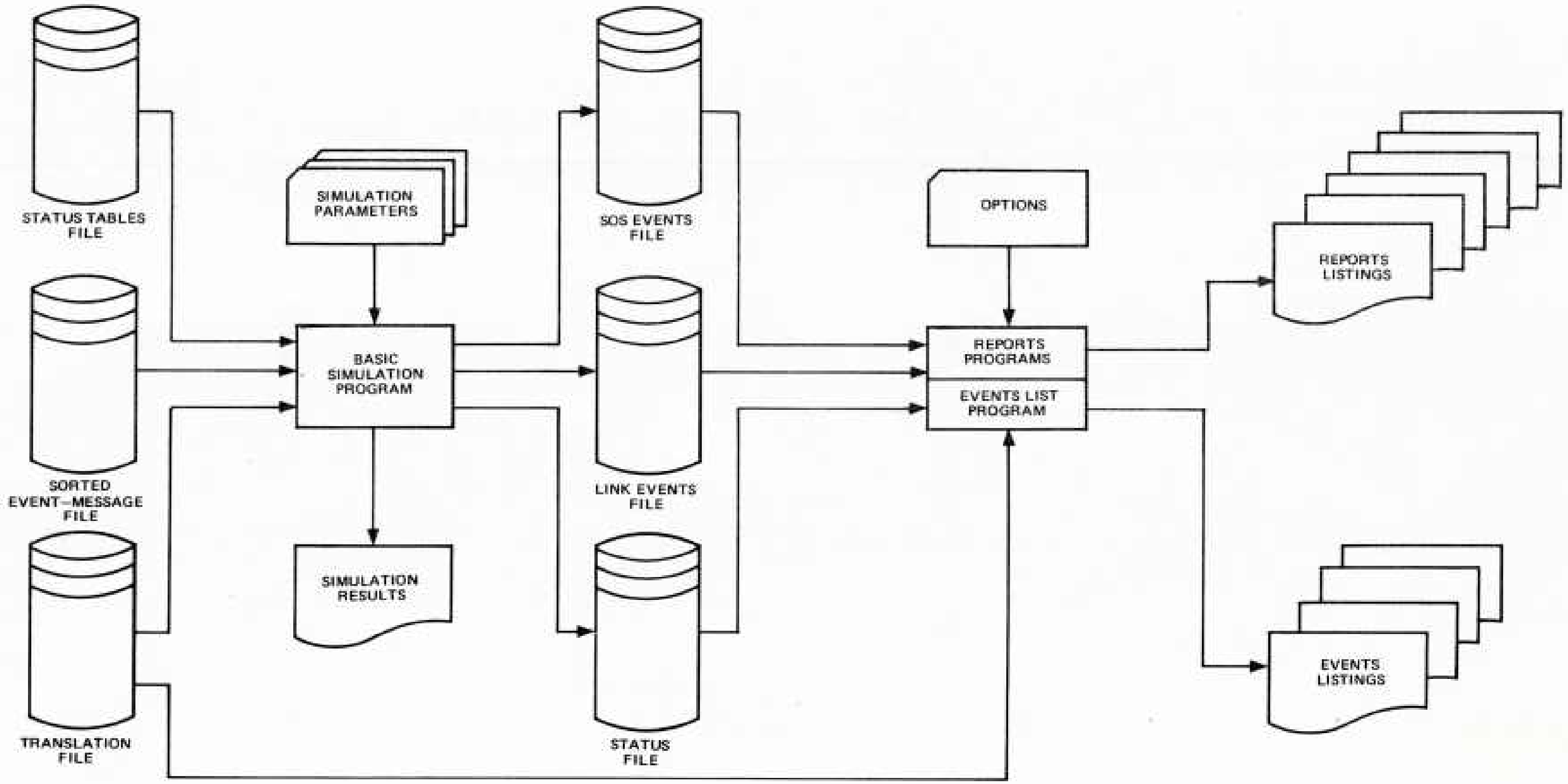


Figure 1-1. Simulation Section Data Flow (Sheet 2 of 2)

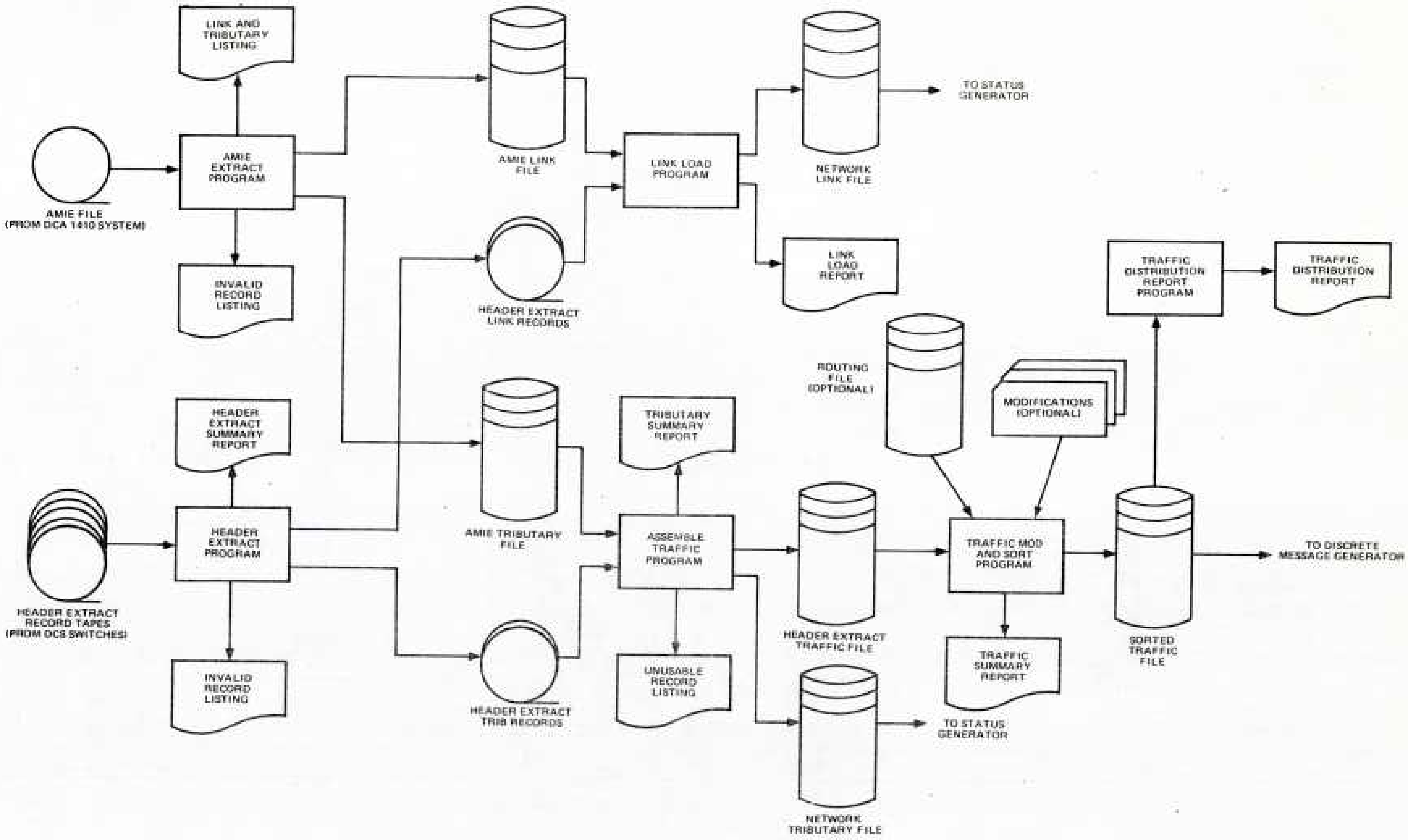


Figure 1-2. Traffic Preprocessing Section Data Flow

SECTION 2 - STATUS GENERATOR PROGRAM

2.1 PROGRAM REQUIREMENT DESCRIPTION

2.1.1 Program Name Status Generator

2.1.2 Program Identification DNSTGEN 1

2.1.3 Purpose

The Status Generator program creates a set of Status Tables (used to drive the Basic Simulation routine) and a Translation Table (used by several programs in the DIN Simulator) for a set of input files which describe a specific AUTODIN network.

2.1.4 Requirements

The program is written in FORTRAN IV programming language. The coding techniques used were as generalized as possible to make the program machine independent. The program was initially implemented on an IBM Model 360/65 and requires approximately 230K bytes of memory for execution.

2.2 INPUT SPECIFICATIONS

Five card-image files are required as input to the Status Generator. These and the Network Configuration, Component, Link, Tributary, and Routing files. The program is designed to allow an entire network to be defined in the Component, Link, Tributary, and Routing files, but actually use only a subset of the full network for a given simulation run.

The Network Configuration file contains the switch names to be used in a simulation. The Component file contains the names and descriptive data for all of the switches in the network. The program ignores any switches in the Component file which do not appear in the Network Configuration file.

The Link file contains the names and descriptive data for the interconnections (links) between switches. The Tributary file contains the names and descriptive data for the tributaries or subscribers at each switch. The Routing file provides the routing logic for messages being sent from switch to switch. As with the Component file, only those items pertaining to the switches given in the Network Configuration file are used.

2.2.1 General

The following general rules apply when filling data fields for the input files to the Status Generator program. All fields are either name fields, numeric data fields, or blank.

Names may be any combination of alphabetic characters (A-Z) or numeric characters (0-9). Special characters ($, &, etc.) are not allowed. A name may be entirely numeric. Names should be left-justified in their fields. Trailing blanks are acceptable, but leading or imbedded blanks are treated as errors.

Numeric data items should be right-justified in their fields. Leading zeroes are not required, but are acceptable to the program. Trailing blanks may not be used in lieu of zeroes, and are ignored by the program. All numeric data fields are assumed to be positive integers and decimal points and signs (+ or -) are treated as errors. A data field of all blanks is treated as zero.

Blank data fields should be blank. Nonblank characters appearing in these fields cause warning messages to be generated by the program on the assumption that they constitute probable keypunch errors.

2.2.2 Network Configuration File

This file contains the names of switches or components which will be extracted from the Component file and used in a simulation. The names are given one per card as described in Table 2-1.

2.2.3 Component File

To effectively simulate a store-and-forward network the characteristics of the switching centers must be adequately described, and, since these centers may not be identical in all cases, provision must be made to supply the characteristics of each center.

A basic switching center has at least four functions to perform: collecting incoming messages, processing incoming messages, scheduling outgoing messages, and processing outgoing messages. The switching center may be as simple as a torn tape relay where the time to perform these functions is measured in minutes and seconds or it may be a complex computer system with measurements in microseconds or milliseconds. Given an input which describes the time required to perform each of the functions and/or subfunctions, plus other inputs which describe the size of the centers and network involved, almost any store-and-forward network can be effectively simulated.

The purpose of the Component file is to describe the size of each of the centers and the time required to perform its functions. The input is arranged so that a single overall time may be given for each function regardless of the volume of transactions, or individual times may be specified for each subfunction and the overall time may be accumulated based on the volume of transactions. Thus, by proper selection of characteristics, a center may be operated in either a synchronous or nonsynchronous mode.

Each network switch must be described by a set of component cards; one to describe input, one to describe output, and one to describe the LTC or ADU data transfer units. The formats of these cards are given in Tables 2-2 through 2-4.

2.2.4 Link Input File

A link is described as a channel or group of channels which connect two switches. The Link file is a group of card sets which define the characteristics of the network links being simulated. Each set describes a single link and may be composed of one or more cards. If a link is composed of one or more channels, all operating at the same transmission rate, the link may be described by a single card. If the link is composed of two or more channels operating at different transmission rates, a separate card is required for each of the transmission rates. All links are considered bidirectional; that is, they can transmit messages from both switches. When a particular simulation problem demands unidirectional links, the link name should only appear on the routing table associated with the sending switch. The format of this card is given in Table 2-5.

2.2.5 Tributary Input File

A tributary is described as a channel or group of channels which connect a subscriber to the AUTODIN network. Messages enter the network from tributary input channels and final message destination delivery is made on the tributary output channels. The Tributary file is a group of tributary input cards which define the network subscribers. Only one card is required to describe each tributary. The format of this card is given in Table 2-6.

2.2.6 Routing File

To transmit a message from a tributary to one located at a distant switch, the simulator must be provided with some form of routing guide. This is the purpose of the Routing file. Within the DIN Simulator, routing is performed for each message after it arrives at a switch. The choice of outgoing link is made relative to the switch at which the message is currently being processed, as opposed to the decision being made relative to the originating switch. Accordingly, each switch must independently route messages to every other switch in the network. This requires a separate table to be supplied for each switch in the network. When the message is to be routed, the destination switch code is used to index the table associated with the current switch to determine which link will be used to transmit the message.

Each network switch must have a routing table in which a transmission path is given to every other switch in the network. Each path between a pair of switches is described in a routing card. The format of the card is given in Table 2-7. One card must be supplied for every pair of switches in the network. This applies in both directions. That is, a routing card from Switch A to Switch B, and one from Switch B to Switch A must be given. A direct link between switches need not be defined. If it is necessary to go through Switch C to get from Switch A to Switch B, the routing card for Switch A to Switch B would specify a link from Switch A to Switch C. The total number of routing cards required is N times (N-1), where N is the number of switches in the network.

2.3 OUTPUT SPECIFICATION

Two output files are generated by the Status Generator. They are the Status Table file used to drive the basic simulation program, and the Translation file which cross-references user names for switches, links, and tributaries with Status Table addresses for these items.

2.3.1 Status Table File

The Status Table file contains information describing the network defined by the input files. This information is passed to the Basic Simulation program and is used by that program in the network simulation.

To provide flexibility in the size and structure of the Status Tables, a special FORTRAN Preprocessor program (CSCSCRPT) is used. The code written for the Status Generator uses pseudo-instructions and mnemonic names rather than actual displacements within the Status Table to fill data items. The Preprocessor program interprets these pseudo-instructions and generates the FORTRAN coding necessary to place the data items in their correct positions within the table. Thus, the makeup of the Status Tables can be changed without changing the code within the Status Generator program. This process is more fully described in Paragraph 2.4.

The actual output file is created by writing the FORTRAN arrays in binary format. The blocking factors for the file are specified at run time via Job Control Language (JCL). The two arrays are NODRAY (the Node Status Table) and STORAY (the Link and Tributary Status Table). Their sizes are determined via CSCSCRPT table entries. The Basic Simulation program uses these same CSCSCRPT tables and is thus able to correctly read in the Status Tables and properly reference the data contained in them. The actual structure of the Status Tables is described in Appendix D.

2.3.2 Translation File

The Translation file is also created from the input files. The most important function of this file is to relate switch, link, and tributary names to their addresses in the Status Tables. This is necessary because the actual names are not passed in the Status Tables to Basic Simulation. The output from the Basic Simulation program is in terms of addresses rather than names, and the Translation file is used as a device to enable this information to be displayed in understandable form. In addition to its translation function, this file is used to pass many of the numeric data items found in the Link and Tributary Input files onto other processing programs within the model structure.

The Translation file is written in card image format, and is logically divided into Component, Link, and Tributary sections. Component cards are identified by a C in card column (CC) 1. Regardless of their order in the Network Configuration file, the component cards appear in alphabetically sorted order in the Translation file.

Immediately following the Component cards are the Link cards. They are identified by an L in CC 1. These Link cards differ from the Link cards in the Link Input file because they are one-directional. For each input Link card describing a two-directional connection between a pair of switches, two output Link cards are generated, one for each direction. These cards are arranged by source switch name (in the same order as the Component cards) and are further sorted alphabetically by link name at each switch.

Following the Link cards are the Tributary cards, identified by a T in CC 1. These are generated on a one-for-one basis from the Tributary Input cards. Like the Link cards, they are sorted by component or switch name, and then by tributary name at each switch.

All of these cards are written in one generalized format, given in Table 2-8. Each card type uses a subset of this general format for its own fields, but does not use the same subset as the other card types. Tables 2-9 through 2-11 give the format and content of the Component, Link, and Tributary cards, respectively. Alphabetic fields are left-justified with trailing blanks, and numeric fields are right-justified, with any leading zeroes replaced by blanks. A zero-valued field has a single zero in the right-most digit position.

In addition to the Status Table and Translation file, a listing is made of each of the input files, along with any error diagnostic messages relating to that file. If there were no serious input errors, a storage utilization map, showing the storage requirements of the network, is printed.

2.4 PROGRAM LOGIC

The program flow of the Status Generator can be divided into six distinct steps. The first five involve reading and processing each of the five input files. The last step creates the Status and Translation files, given that the input was free of serious errors.

2.4.1 Network Configuration File

Each card from the file is read into core in 80A1 format and put into array NCARD (80). Subroutine TRANSL is called to verify that the name is syntactically correct and to translate the six-character name into a unique integer value (one which collates exactly as the original name would). The input card, followed by any error diagnostic message(s), is printed; the character version of the name is moved into array NCOMP (6, I); and the integer version of the name placed into array COMPT (I).

After all cards have been read, the names in both arrays are sorted in alphabetical order using subroutines CHNSRT and DCHAIN. A check is then made for duplicate switch names. If any are found, the extra names are removed and error messages printed.

Subroutine SUMMRY is called to print out an error summary report, and control is passed to the Component file.

2.4.3 Component File

Each card is read into array NCARD (80), and subroutine TRANSL is called to translate the switch name. If the switch name is invalid, the card is printed with an error diagnostic and the next card is read. If the name is valid, a search is made of the translated switch names in array COMPT. If a match is not found, the component is not in this particular network configuration and the card is ignored. The unique names of all such ignored components are kept in arrays XNODES (I) and XTRA (6, I).

If the switch is in the network, the input card is kept. A check is made of blank fields for nonblank characters, with a warning printed for each. Next, a search is made for card type; i.e., INPUT, OUTPUT, LTC/ADU. If the type cannot be identified, the remaining data fields cannot be checked and processing goes to the next card.

The input card is printed and the remaining data fields are converted using subroutine CONVRT. Each field is checked for being within its allowable range. The data for each component is put into array MCOMP (15, I) in the same relative position as the switch name in array COMPT (I).

When all cards have been read in, a check is made for missing cards, and an error message written for each missing card. Subroutine SUMMRY is called and control passed to the Link file.

2.4.3 Link File

Each card is read into core and put into array NCARD (80). Subroutine TRANSL is called twice to translate both switch names. If either switch name is invalid, processing goes to the next card. If the names are syntactically valid, a search is made in array COMPT for each switch name to determine whether it is included in the network. Only if both switches are in the network is the card kept, otherwise an additional search is made in array XNODES for a match. If there is still no match, the name is in error.

After verifying switch names, the remainder of the card is checked for errors. The link name is translated, and the translated name is put into the temporary variable X. The alphabetic link name is copied into the first six words of temporary array NFIELD (12). The data fields are converted, checked for being in allowable range, and put into the remaining six words of NFIELD. Small data items are packed two to a word to conserve storage. If there was any error in the card, it is ignored.

If the card is error-free, the variable X and array NFIELD are written in card-image form onto work file NWRK3. The switches are then interchanged and another record is written which defines a link with the source and destination switches reversed. A count is kept of how many link cards are written for each switch as a source.

After all cards have been processed, file NWRK3 is rewound and the data is read back into core. The translated links names are put into array XNAMES (I) and the data put into array IWORK (8,I). The alphabetic names are now packed in A4, A2 format, which reduces the storage required for each link card from 12 to 8 words. Since it is known how many links are associated with each source switch, it is possible to group them by source switches in arrays XNAMES and IWORK as they are read in. Using subroutines CHNSRT and DCHAIN, the link names and data are sorted in alphabetical order within each source switch.

At this point a number of overall error checks are made. Warnings are posted for switches which have zero or only one link. Each link (which may consist of multiple link cards) is checked to ensure that the total number of channels do not exceed 255. Also, warnings are issued if the security indicators and/or queue delays are inconsistent across multiple link cards defining a single link. Several different checks are made to ensure uniqueness of link names between switch pairs.

Finally, subroutine SUMMRY is called and control passes to the Tributary file.

2.4.4 Tributary File

Each card from the file is read into array NCARD. The switch name of the switch servicing the tributary is checked for inclusion in the network. If it is not in the network, the card is ignored.

The tributary name is translated and the integer value placed in variable X. The remaining data fields are converted, checked for being acceptable values, and put into the last eight words of array KFIELD (14). If there were any errors on the card processing goes to the next card, otherwise, the alphabetic tributary name is copied into the first six words of array KFIELD, and variable X and array KFIELD are written into work file NWRK3.

When all input cards have been handled, the data is read back into core into array JWORK (10, I). As for the Link file, the data is sorted by switch and then by tributary name within each switch. A check is now made for duplicate tributary names and for switches which have no links and no tributaries. Subroutine SUMMRY is then called to print a summary of errors for the file.

2.4.5 Routing File

Each card from the Routing file is read into core and the source and destination switch names are translated and checked for inclusion in the network. If both switches match, the card is kept. Next, the primary and alternate (if any) link names are translated. A search is made through array XNAMES to find the link(s). If the link name is not found the card is skipped.

The position in array XNAMES of the primary link is stored in NROUT (1, I) and the alternate link position in NROUT (2, I). If there is no alternate link, a zero is inserted. Array NROUT is N times N words in length, where N is the number of switches in the network. The position in the table for the data from a given routing card is determined by the source and destination switches. If the source switch is position I in array COMPT and the destination switch is position J, the link data are put into position N*(I-1) +J in the routing array.

When the entire file has been processed, a check is made for missing cards. If all routing pairs have been supplied, the paths from each switch to every other switch are traced to ensure that the routing has been properly defined. Subroutine SUMMRY is called to complete the file processing.

2.4.6 Status Table and Translation File Generation

The first action taken in this section of the program is to determine whether serious errors were encountered in any of the input files. If there were, the Status Table and Translation files are not created, and the program terminates with a completion code of 16.

Next, a check as to how much storage the Link and Tributary Status Table requires is made in the following manner. A CSCSCRPT statement, P INSERT SIZES, causes a FORTRAN DATA statement to be inserted into the program. This statement defines the number of words required in the table for each link, link channel, tributary, tributary channel, and route. Other sizes are also included by this statement.

The total necessary storage is computed as:

```
  LINKSZ * Number of Links
+ LINKCH * Number of Link Channels
+ TRIBSZ * Number of Tributaries
+ TRIBCH * Number of Tributary Channels
+ RTSIZE * (Number of Switches) **2
```

Array STORAY contains the Link and Tributary Status Table, and its size is found in variable ENDCOR. Since the table size can be varied via CSCSCRPT, and the network size can be varied by the user, it is necessary to ensure that sufficient storage is available to simulate the network. If the network requirement is larger than ENDCOR, the program terminates with a completion code of 16.

To conserve storage, array JWORK, which contains the input data from the Tributary file, is equivalenced to array STORAY. Accordingly, array JWORK is now written out onto work file NWRK 12. It is later read back in one tributary at a time. The Node Status Table (array NODRAY (I, J)) and the Link and Tributary Status Table are set to zero.

Array NODRAY has one row for each switch in the network. It is fixed in size and large enough to accommodate the largest anticipated AUTODIN network. The address for each switch in the Translation Table is its position in NODRAY, which is the same position as in array MCOMP. Since the switch addresses are known in advance, the component portion of the Translation file is written onto file NTRANS.

The Status Tables are now filled in the following manner. Variable NODNO is used to point to the location in NODRAY of the switch being worked on. Its value starts at 1 and is incremented by 1 as each switch is processed into the table. Variable LNKADD points to the next available word in STORAY. Its value starts at 11 and is incremented by the appropriate number of words as each link or tributary is added.

Processing begins with the first switch in MCOMP. The first action is to copy the component data from MCOMP to NODRAY. This is done via CSCSCRPT MOVE instructions. This allows coding source program statements to fill the table without the programmer needing to know exactly where each field is to go. Considerable flexibility is imparted by this technique, since the Status Table format can be changed without any programming changes being necessary.

Those fields in the Status Tables not filled by the Status Generator are later filled by the Basic Simulation program. Each specific field is assigned a unique mnemonic name. CSCSCRPT is used to related these names to specific words or parts of words within the Status Tables. The Node Status fields filled by the Status Generator are:

NSFL, NSNT, NSNLT, MPLTCD, MPNLTC, MPLTCX, MPIPID, MPISOM, MPIRIL, MPIRIH, MPOPOD, MPOSOM, MPOMX, MPOFQ, MPNFQ, MPMISC, FETT, FETU, FELBA1, FELBA2, FELBA3, FELBA4.

The mnemonic names and meanings of all fields in the Node, Link and Tributary Status Tables are given in Tables 6-7 through 6-9 in Section 6.

The information for each of the links at this switch is added to the Link and Tributary Status Table. The data fields from each link card are extracted from array IWORK (8, I), unpacked, and put into temporary array LF (9). If this is the first link card for the link, the following basic link fields are filled:

LTNC, LTCA, LTLTC, LTSN, LTSC, LTQS, LTPRE, LNDN

Room is then left in the table for each channel described by this card, and link channel fields LCSP and LCAD are filled. If this is not the first link card for the link, only the channel items are filled.

The link address is the starting point in STORAY for the link data. Since the link address is known, the Translation file entry for this link card is written out onto file TRANSL. The identical link address is used for all link cards which define a single link.

After all links have been processed, the tributary data for all tributaries at the switch are added to the Status Table immediately following the link information. Each tributary's data is read into core, unpacked, and put into temporary array JF (10). The following tributary items are then filled in the Status Table:

LTNC, LTLTC, LTSN, LTTN, LTSC, LTCA,
LTQS, LTPRE, TRVN, TRSPT, TRSPC, TRSPM, TRAD

Room is left for each tributary channel, but no fields are filled in them. The Translation file entry for the tributary is written on temporary file NWRK11.

The preceding steps are repeated until all switches, links, and tributaries have been processed. At this point file NWRK11 is rewound and copied onto file NTRANS to complete the Translation file.

The routing data is then added to the Status Table, starting at the next available location. The starting point for routes for each switch is saved in the Node Status Table in field NSRT. The routing input data contained in array NROUT (2, I) is copied into the Status Table, where link positions are changed to link addresses. Field mnemonics RTP and RTS are used.

Any remaining space in STORAY is chained together in units as defined by the SIZES DATA statement. This area is used as dynamic storage by Basic Simulation.

Finally, the Status Tables are written in binary form onto file NSTAT. If any warning messages were generated during the input file processing, the program terminates with a completion code of 4; otherwise the program terminates normally.

2.4.7 Subroutine TRANSL

This routine translates six-character names into integer numbers. The technique ensures that the translated numbers are unique for each name and will collate in the same manner as the original name. This offers a considerable advantage in a FORTRAN program which must do large scale manipulation with names, since each six-character name can be treated as an integer variable, thus reducing program coding and running time.

The subroutine recognizes 37 valid characters, each of which is assigned a numeric value from 0 to 36. The characters are blank, the letters A-Z, and the digits 0-9. Numeric values are assigned in ascending order. Thus a blank receives a value of 0, A is assigned a value of 1, Z gets a value of 26, the digit 0 is a 27, and digit 9 receives a value of 36.

The name is translated by taking each character in turn from left to right, multiplying it by the appropriate power of 37 and adding it to the value, which is initialized at - 2147483647 (reflecting that initial implementation was on an IBM/ 360). More specifically, the value is computed as:

-2147483647

+ 1st character * 37**5

+ 2nd character * 37**4

+ 3rd character * 37**3

+ 4th character * 37**2

+ 5th character * 37**1

+ 6th character * 37**0

Consider the name "ABCZ19." It recieves a translated value of:

-2147483647

+ 1 * 37**5

+ 2 * 37**4

+ 3 * 37**3

+26 * 37**2

+27 * 37

+37

This number is unique from the number assigned to any other name. Any name which would come before "ABCZ19" in an alphabetically sorted sequence has a lesser numeric value assigned to it. For instance, the name "ABCZ18" has a value 1 less than "ABCZ19."

2.4.8 Internal Sorting

Two subroutines, CHNSRT and DCHAIN, are used in the Status Generator to sort names and data. The routines operate first on an array of translated names which are to be sorted. Subroutine CHNSRT, using a double-linked chain, takes each value in turn from the input array, finds its position in the chain, and modifies the chain to record its relative order in the data. At the completion of subroutine CHNSRT, the chain will show, for each name in the input array, what its position should be in the array after sorting. The input array is unchanged.

The data corresponding to the names are in a separate array, where the data items are in the same relative position as the names. Subroutine DCHAIN unravels the chain created by CHNSRT and moves each name and its data into their sorted positions. This is done with a minimum of movement, with each item at most placed once into a temporary work area prior to being put in its final location.

These routines were written to take advantage of the assumption that the data would be already partially ordered on input. If the assumption is untrue, more computer time will be used with more simple sorting techniques.

Program logic flow charts are included at the end of this section.

2.5 MACHINE DEPENDENCE

The Status Generator program is almost entirely machine independent except for possible minor compiler variations between machines. The only known restrictions are that the word size of the executing computer must be 32 bits or larger, and a word must hold four characters or more of the alphabetic data.

Table 2-1. Network Configuration File

CARD COLUMN	DESCRIPTION	EXPLANATION
1-6	Switch Name	Model name of a switch to be included in the simulation.
7-80	Blank	

Table 2-2. Switch Input Card

CARD COLUMN	DESCRIPTION	EXPLANATION
1-6	Switch Name	Model name for the switching center.
7	Blank	
8-12	Card Type	INPUT
13-18	Blank	
19-24	OH Delay for PID	Overhead time in milliseconds to process input data. Value can be 0-600000.
25	Blank	
26-31	Message Processing Delay	Time in milliseconds to process each message. Value can be 0-600000.
32	Blank	
33-38	RI Delay (Low precedence)	Time in milliseconds to process each low precedence message through the routing programs. Value can be 0 - 600000.
39	Blank	
40-45	RI Delay (High precedence)	Time in milliseconds to process each high precedence message through the routing programs. Value can be 0 - 600000.
46-80	Blank	

Table 2-3. Switch Output Card

CARD COLUMN	DESCRIPTION	EXPLANATION
1-6	Switch Name	Model name for the switching center.
7	Blank	
8-13	Card Type	OUTPUT
14-18	Blank	
19-24	OH Delay for POD	Overhead time in milliseconds to process Output Data. Value can be 0 - 600000.
25	Blank	
26-31	Message Processing Delay	Time in milliseconds to process each outgoing message. Value can be 0 - 600000.
32	Blank	
33-38	Message Exchange Delay	Time in microseconds per line block to convert message from one format to another. Value can be 0 - 600000.
39	Blank	
40-45	Fixed Queue	Maximum number of messages in queue at a switch at one time. If zero or blank, no limit is applied. Value can be 0 - 16383.
46-80	Blank	

Table 2-4. Switch LTC/ADU Card

CARD COLUMN	DESCRIPTION	EXPLANATION
1-6	Switch Name	Model name for the switching center.
7	Blank	
8-14	Card Type	LTC/ADU
15-23	Blank	
24	Number of LTC/ADUs at Switch	From 1 to 4
25	Blank	
26-31	Line blocks available	Line block size of each LTC or ADU. Value can be 0 - 32767*
32	Blank	
33-38	Throttle Line Blocks	Point at which the LTC/ ADU must stop accepting input. (maximum 32767)*
39	Blank	
40-45	Unthrottle Line Blocks	Point at which the LTC/ADU may resume accepting inputs. (maximum 32767)*
46	Blank	
47-52	Transfer Delay	Time in microseconds to transfer one line block to or from an LTC/ADU. Value can be 0 - 600000.
53	Blank	
54-59	Overhead LTC Delay	Time in milliseconds to read or write on LTC/ADU. Value can be 0 - 600000.
60	Blank	
61-66	MISC Overhead Delay	Miscellaneous cycle delay in milliseconds. Value can be 0 - 600000.
67-80	Blank	

*Line Blocks Available, Throttle and Unthrottle Line Blocks may all be zero indicating that LTC/ADU function is not to be simulated. If Line Blocks Available is other than zero, values must also be specified for Throttle and Unthrottle Line Blocks. Throttle Line Blocks should not be less than 512. Unthrottle Line Blocks must be larger than the Throttle value.

Table 2-5. Link Input Card Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-9	Blank	
10-15	Link Name	Model identification for the link during simulation time.
16	Blank	
17-19	Number of Channels	Number of channels operating at this transmission rate in the link. Total must not exceed 255.

21	Security Indicator	Highest security classification of traffic allowed on the link.
		U = Unclassified
		C = Confidential
		S = Secret
		T = Top Secret.
		A = Special

22-24	Blank	
25-29	Transmission Delay	Time in milliseconds to transmit one line block over the link. Value can be 1 - 32767.
30	Blank	
31-35	End of Message Acknowledgement	Time in milliseconds for acknowledgement of transmitted message to be receipted by the destination switch. Value can be 0 - 32767.
36	Blank	
37-42	Switch Name 1	Model name for switch servicing the link.
43	Blank	
44	LTC Reference 1	LTC that services the link at switch specified in Switch Name 1.
45	Blank	
46-51	Switch Name 2	Model name for the second switch servicing link.
52	Blank	
53	LTC Reference 2	LTC that services the link at switch specified in Switch Name 2.
54	Blank	
55-59	Queue Delay	Average starting queue time for low priority messages in seconds (may be blank). For messages going from Switch 1 to Switch 2, value can be 0 - 32767.
60	Blank	

Table 2-5. Link Input Card Format (Cont'd)

CARD COLUMN	DESCRIPTION	EXPLANATION
61-65	Queue Delay	Same as above, but for messages going from Switch 2 to Switch 1.
66-80	Blank	

Table 2-6. Tributary Card Input Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-9	Blank	
10-15	Tributary Name	Model tributary identification
16	Blank	
17-19	Number of Channels	Channels at the tributary. Maximum value is 255.
20	Blank	

21	Security Indicator	Specifies highest security classification of traffic on tributary channels.
		U = Unclassified
		C = Confidential
		S = Secret
		T = Top Secret
		A = Special

22-24	Blank	
25-29	Teletype Transmission Delay	Time in milliseconds to transmit one line block of teletype over the tributary's channels. Value can be 1 - 32767.
30	Blank	
31-35	Punched Card Transmission Delay	Time in milliseconds required to transmit one line block of a punched card message* (maximum 32767)
36	Blank	
37-41	Magnetic Tape Transmission Delay	Time in milliseconds required to transmit one line block of a magnetic tape message. * (maximum 32767)
42	Blank	
43-47	End of Message Acknowledgement	Time in milliseconds for message acknowledgement before next message can begin. * (maximum 32767)
48	Blank	
49-54	Switch Name	Model name for the switch servicing the tributary.
55	Blank	
56	LTC Reference	Number of the LTC/ADU that services the channel(s) for the tributary.
57	Blank	
58-62	AUTOVON Dial Delay	If the tributary is serviced through AUTOVON, time in seconds required to dial a line* (maximum 16383 seconds)
63	Blank	

Table 2-6. Tributary Card Input Format (Cont'd)

CARD COLUMN	DESCRIPTION	EXPLANATION
64-68	Queue Delay	Average starting queue time for low priority messages.* (In seconds) (maximum 65535)
69-80	Blank	

*These fields may be blank.

Table 2-7. Routing Card Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-6	Source Switch	Model identification of the source switch.
7-9	Blank	
10-15	Destination Switch	Model identification of the destination switch.
16	Blank	
17-22	Primary Link Name	Name of link that is primary route for all message transmissions to switch specified in the Destination Switch Name of the card. *
23	Blank	
24-29	Secondary Link Name	Name of link that is secondary route (as above). * May be blank.
30-80	Blank	

*One end of the link described must terminate at the source node.

Table 2-8. Translation File - General Format

CARD COLUMN	DESCRIPTION	EXPLANATION

1	Alphabetic*	Identifier:
		C = Component
		L = Link
		T = Tributary

2-7	Alphabetic	Component, link, or tributary name
8	Blank	
9-14	Numeric	Component, link, or tributary Address
15	Blank	
16-21	Alphabetic	**
22	Alphabetic	
23-28	Alphabetic	
29	Blank	
30-31	Numeric	
32	Blank	
33-34	Numeric	
35	Blank	
36	Numeric	
37	Blank	
38	Numeric	
39	Blank	
40-42	Numeric	
43	Blank	
44	Alphabetic	
45	Blank	
46-50	Numeric	
51	Blank	
52-56	Numeric	
57	Blank	
58-62	Numeric	
63	Blank	
64-68	Numeric	

Table 2-8. Translation File - General Format (Con't)

CARD COLUMN	DESCRIPTION	EXPLANATION
69	Blank	
70-74	Numeric	
75	Blank	
76-80	Numeric	

*Alphabetic means here the letters of the alphabet and/or the digits 0-9.

**The remaining fields vary for each type.

Table 2-9. Translation File - Component Cards

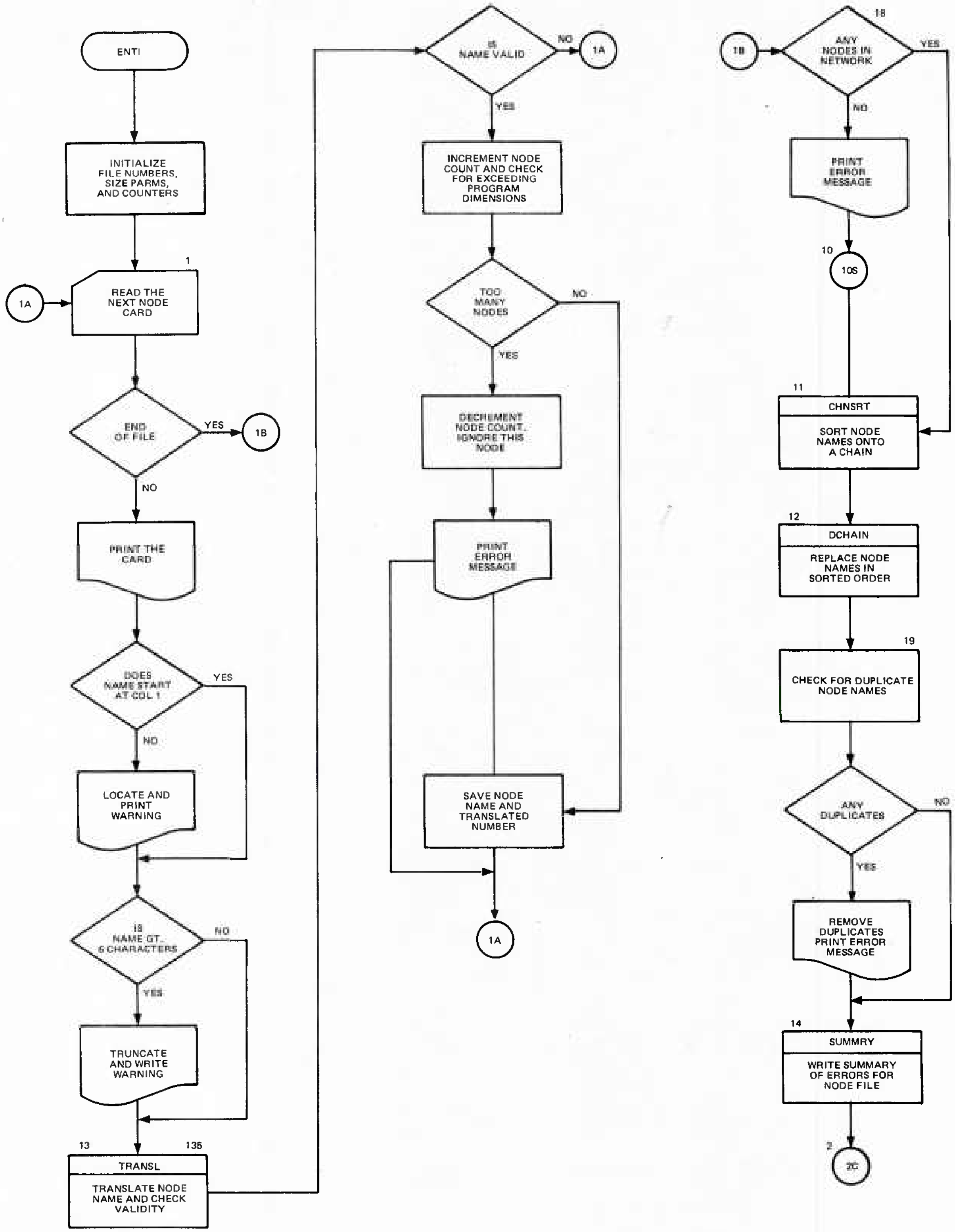
CARD COLUMN	DESCRIPTION	EXPLANATION
1	C	Identifier
2-7	Component name	Name for the Switching Center
8	Blank	
9-14	Address	Location in Switch Status Table
15-45	Blank	
46-50	Number of Links	Number of link cards (not links) for this switch in the Translation file.
51	Blank	
52-56	Number of Tributaries	Number of Tributary cards for this switch in the Translation file.
57-80	Blank	

Table 2-10. Translation File - Link Cards

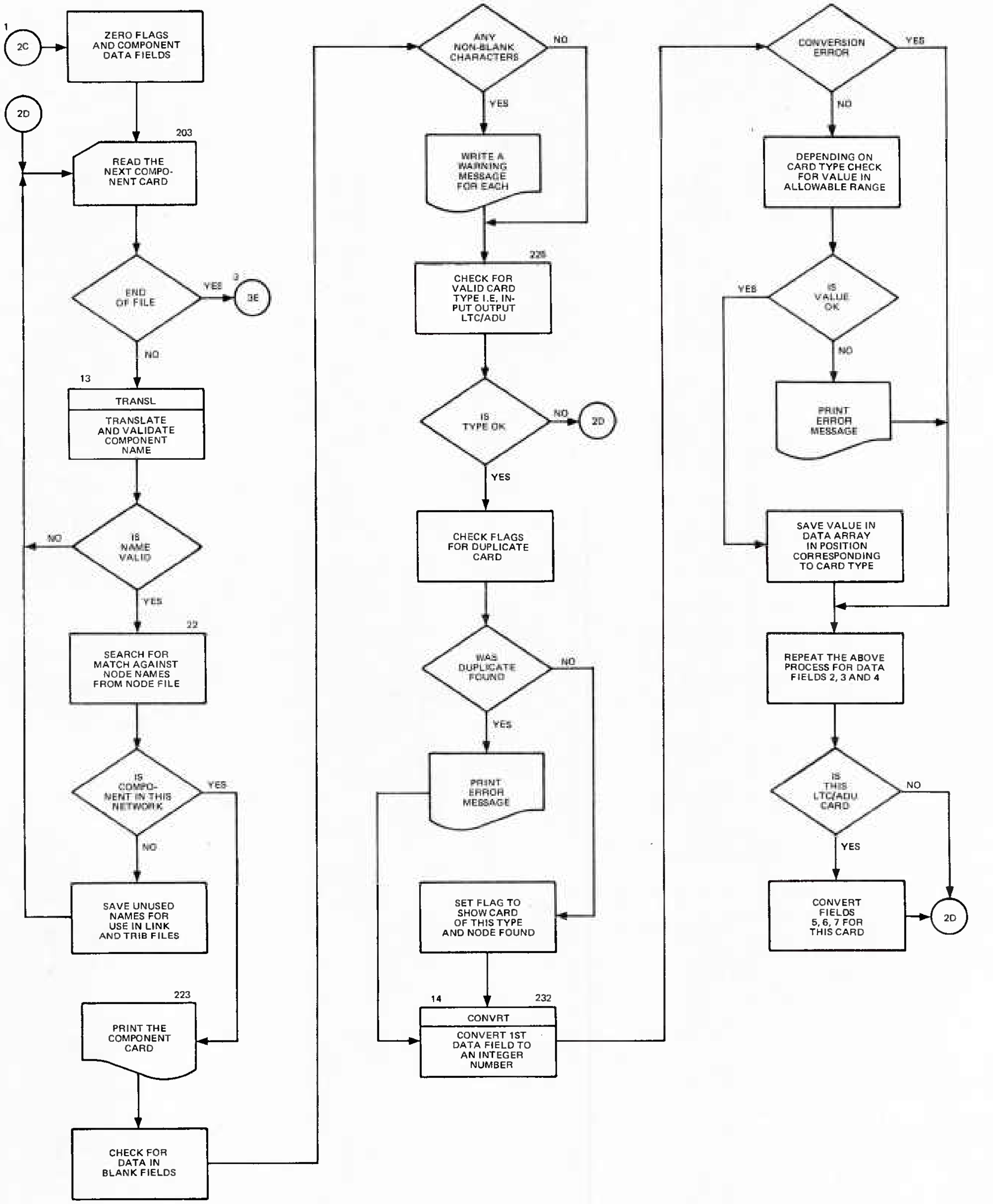
CARD COLUMN	DESCRIPTION	EXPLANATION
1	L	Identifier
2-7	Link Name	Model identification for the link
8	Blank	
9-14	Address	Location of this link (the entire link - not just this card) in the Link and Tributary Status Table.
15	Blank	
16-21	Switch 1 Name	Source switch for this link
22	Blank	
23-28	Switch 2 Name	Destination switch for this link
29	Blank	
30-31	Switch 1 Address	Location of source switch in the Switch Status Table
32	Blank	
33-34	Switch 2 Address	Location of destination switch in the Switch Status Table.
35	Blank	
36	LTC Reference 1	LTC at the source switch which services this link.
37	Blank	
38	LTC Reference 2	LTC at the destination switch which services this link.
39	Blank	
40-42	Number of Channels	Number of channels associated with the link card (not the entire link)
43	Blank	
44	Security Indicator	Security level for the link
45	Blank	
46-50	Transmission Delay	Time in milliseconds to transmit one line block over the link
51	Blank	
52-56	End of Message Acknowledgment Delay	Time in milliseconds required for message acknowledgment
57	Blank	
58-62	Queue Delay	Average starting queue time for low priority messages (in seconds)
63-80	Blank	

Table 2-11. Translate File - Tributary Cards

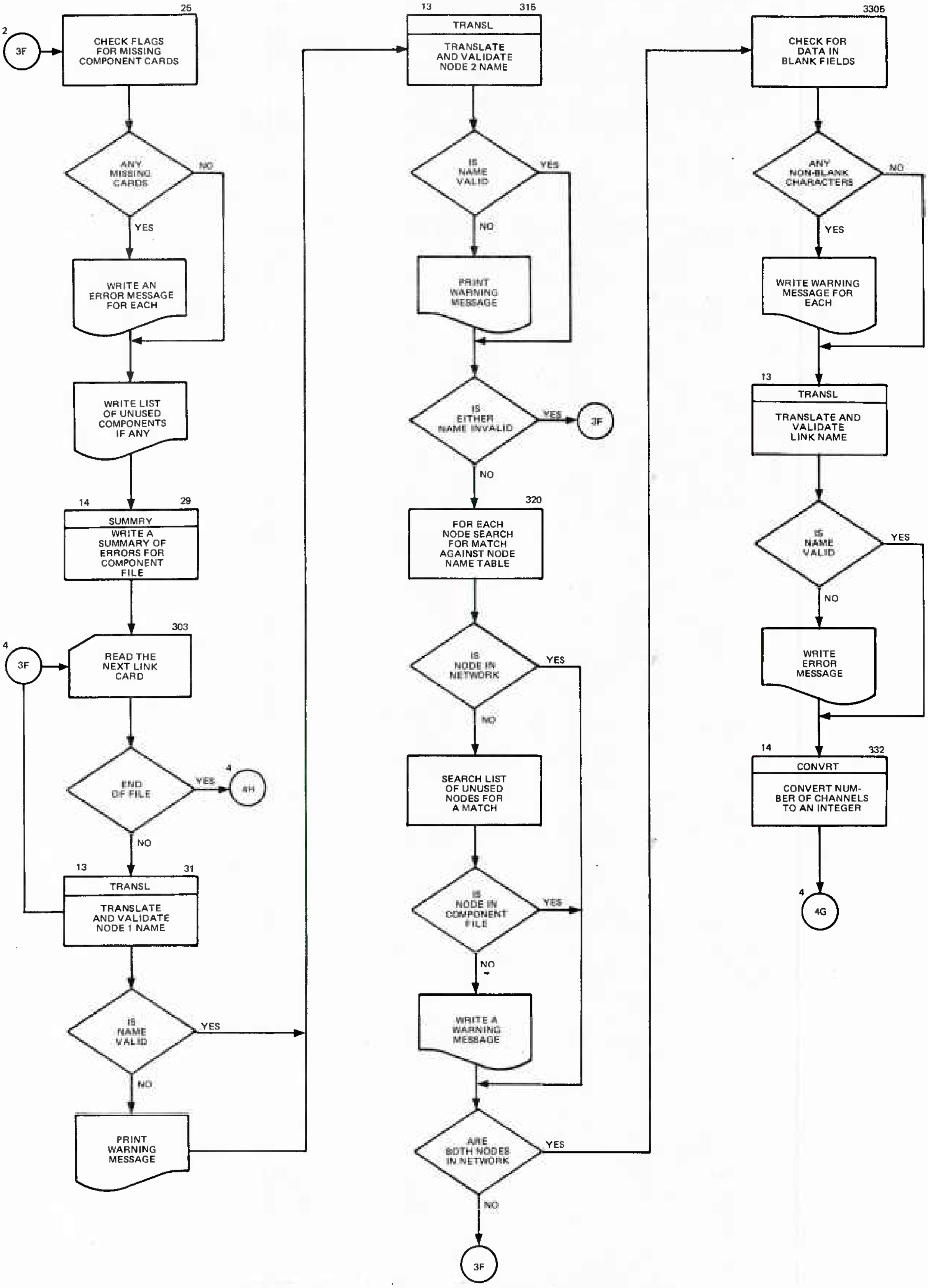
CARD COLUMN	DESCRIPTION	EXPLANATION
1	T	Identifier
2-7	Tributary Name	Tributary identification
8	Blank	
9-14	Address	Location of this tributary in the link and Tributary Status Table.
15	Blank	
16-21	Switch Name	Switch which services this tributary
22-29	Blank	
30-31	Switch Address	Location in the Switch Status Table of the servicing switch.
32-35	Blank	
36	LTC Reference	LTC at above switch which services the tributary.
37-39	Blank	
40-42	Number of Channels	Number of channels associated with this tributary.
43	Blank	
44	Security Indicator	Security level for the tributary
45	Blank	
46-50	Teletype Transmission Delay	Time in milliseconds to transmit one line block of teletype data
51	Blank	
52-56	Punched Card	Time in milliseconds to transmit one line block of punched card data.
57	Blank	
58-62	Magnetic Tape Transmission Delay	Time in milliseconds to transmit one line block of magnetic tape data
63	Blank	
64-68	End of message Acknowledgement Delay	Time in milliseconds required for message acknowledgment.
69	Blank	
70-74	AUTOVON Dial Delay	Time in seconds to dial a line
75	Blank	
76-80	Queue Delay	Average starting queue time for low priority messages.


ENTI
INITIALIZE FILE NUMBERS, SIZE PARMS, AND COUNTERS
1
1A
READ THE NEXT NODE CARD
END OF FILE
YES
1B
NO
PRINT THE CARD
DOES NAME START AT COL 1
YES
NO
LOCATE AND PRINT WARNING
IS NAME GT. 6 CHARACTERS
NO
YES
TRUNCATE AND WRITE WARNING
13
135
TRANSL
TRANSLATE NODE NAME AND CHECK VALIDITY
IS NAME VALID
NO
1A
YES
INCREMENT NODE COUNT AND CHECK FOR EXCEEDING PROGRAM DIMENSIONS
TOO MANY NODES
NO
YES
DECREMENT NODE COUNT. IGNORE THIS NODE
PRINT ERROR MESSAGE
SAVE NODE NAME AND TRANSLATED NUMBER
1A
18
1B
ANY NODES IN NETWORK
YES
NO
PRINT ERROR MESSAGE
10
10S
11
CHNSRT
SORT NODE NAMES ONTO A CHAIN
12
DCHAIN
REPLACE NODE NAMES IN SORTED ORDER
19
CHECK FOR DUPLICATE NODE NAMES
ANY DUPLICATES
NO
YES
REMOVE DUPLICATES PRINT ERROR MESSAGE
14
SUMMRY
WRITE SUMMARY OF ERRORS FOR NODE FILE
2
2C

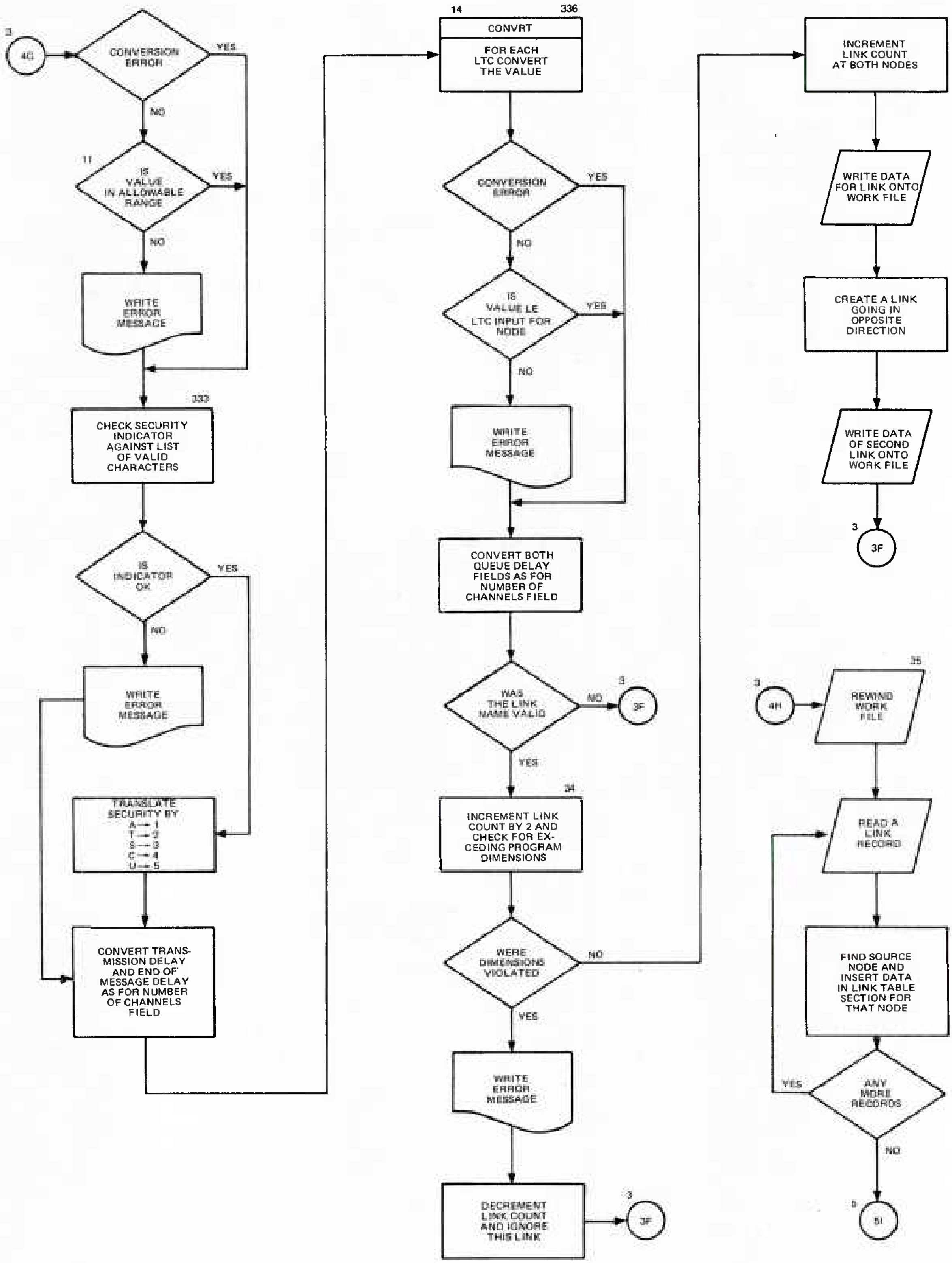
Status Generator Program (Sheet 1 of 14)


1
2C
ZERO FLAGS AND COMPONENT DATA FIELDS
2D
203
READ THE NEXT COMPONENT CARD
END OF FILE
YES
3
3E
NO
13
TRANSL
TRANSLATE AND VALIDATE COMPONENT NAME
NO
IS NAME VALID
YES
22
SEARCH FOR MATCH AGAINST NODE NAMES FROM NODE FILE
IS COMPONENT IN THIS NETWORK
YES
NO
SAVE UNUSED NAMES FOR USE IN LINK AND TRIB FILES
223
PRINT THE COMPONENT CARD
CHECK FOR DATA IN BLANK FIELDS
ANY NON-BLANK CHARACTERS
NO
YES
WRITE A WARNING MESSAGE FOR EACH
228
CHECK FOR VALID CARD TYPE I.E. INPUT OUTPUT LTC/ADU
IS TYPE OK
NO
2D
YES
CHECK FLAGS FOR DUPLICATE CARD
WAS DUPLICATE FOUND
NO
YES
PRINT ERROR MESSAGE
SET FLAG TO SHOW CARD OF THIS TYPE AND NODE FOUND
14
232
CONVRT
CONVERT 1ST DATA FIELD TO AN INTEGER NUMBER
CONVERSION ERROR
YES
NO
DEPENDING ON CARD TYPE CHECK FOR VALUE IN ALLOWABLE RANGE
YES
IS VALUE OK
NO
PRINT ERROR MESSAGE
SAVE VALUE IN DATA ARRAY IN POSITION CORRESPONDING TO CARD TYPE
REPEAT THE ABOVE PROCESS FOR DATA FIELDS 2, 3 AND 4
IS THIS LTC/ADU CARD
NO
YES
CONVERT FIELDS 5, 6, 7 FOR THIS CARD
2D

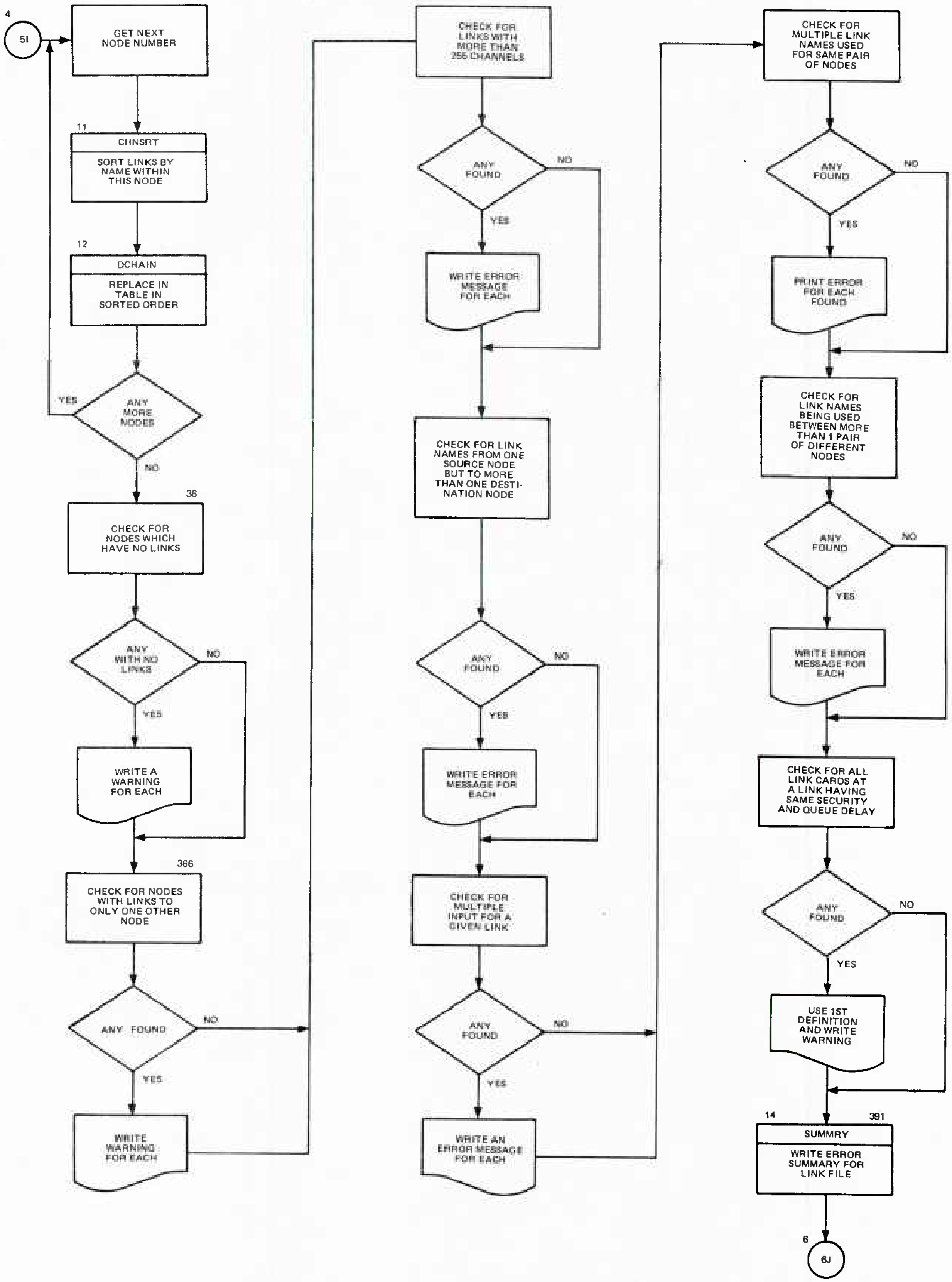
Status Generator Program (Sheet 2 of 14)


2
3F
25
CHECK FLAGS FOR MISSING COMPONENT CARDS
ANY MISSING CARDS
NO
YES
WRITE AN ERROR MESSAGE FOR EACH
WRITE LIST OF UNUSED COMPONENTS IF ANY
14
29
SUMMRY
WRITE A SUMMARY OF ERRORS FOR COMPONENT FILE
4
3F
303
READ THE NEXT LINK CARD
END OF FILE
YES
4
4H
NO
13
31
TRANSL
TRANSLATE AND VALIDATE NODE 1 NAME
IS NAME VALID
YES
NO
PRINT WARNING MESSAGE
13
315
TRANSL
TRANSLATE AND VALIDATE NODE 2 NAME
IS NAME VALID
YES
NO
PRINT WARNING MESSAGE
IS EITHER NAME INVALID
YES
3F
NO
320
FOR EACH NODE SEARCH FOR MATCH AGAINST NODE NAME TABLE
IS NODE IN NETWORK
YES
NO
SEARCH LIST OF UNUSED NODES FOR A MATCH
IS NODE IN COMPONENT FILE
YES
NO
WRITE A WARNING MESSAGE
ARE BOTH NODES IN NETWORK
YES
NO
3F
3305
CHECK FOR DATA IN BLANK FIELDS
ANY NON-BLANK CHARACTERS
NO
YES
WRITE WARNING MESSAGE FOR EACH
13
TRANSL
TRANSLATE AND VALIDATE LINK NAME
IS NAME VALID
YES
NO
WRITE ERROR MESSAGE
14
332
CONVRT
CONVERT NUMBER OF CHANNELS TO AN INTEGER
4
4G

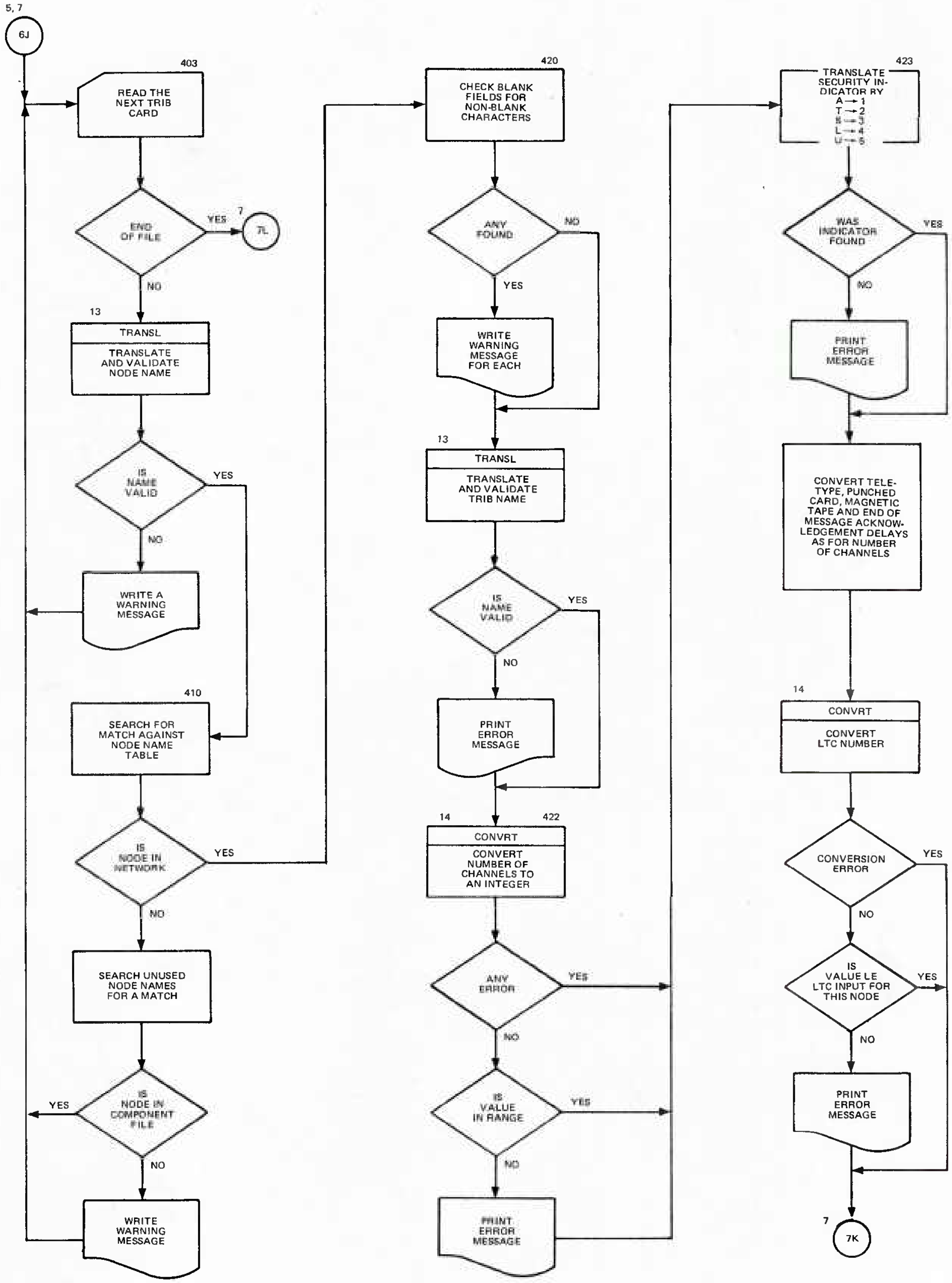
Status Generator Program (Sheet 3 of 14)


3
4G
CONVERSION ERROR
YES
NO
11
IS VALUE IN ALLOWABLE RANGE
YES
NO
WRITE ERROR MESSAGE
333
CHECK SECURITY INDICATOR AGAINST LIST OF VALID CHARACTERS
IS INDICATOR OK
YES
NO
WRITE ERROR MESSAGE
TRANSLATE SECURITY BY
A → 1
T → 2
S → 3
C → 4
U → 5
CONVERT TRANSMISSION DELAY AND END OF MESSAGE DELAY AS FOR NUMBER OF CHANNELS FIELD
14
336
CONVRT
FOR EACH LTC CONVERT THE VALUE
CONVERSION ERROR
YES
NO
IS VALUE LE LTC INPUT FOR NODE
YES
NO
WRITE ERROR MESSAGE
CONVERT BOTH QUEUE DELAY FIELDS AS FOR NUMBER OF CHANNELS FIELD
WAS THE LINK NAME VALID
NO
3
3F
YES
34
INCREMENT LINK COUNT BY 2 AND CHECK FOR EXCEDING PROGRAM DIMENSIONS
WERE DIMENSIONS VIOLATED
NO
YES
WRITE ERROR MESSAGE
DECREMENT LINK COUNT AND IGNORE THIS LINK
3
3F
INCREMENT LINK COUNT AT BOTH NODES
WRITE DATA FOR LINK ONTO WORK FILE
CREATE A LINK GOING IN OPPOSITE DIRECTION
WRITE DATA OF SECOND LINK ONTO WORK FILE
3
3F
3
4H
35
REWIND WORK FILE
READ A LINK RECORD
FIND SOURCE NODE AND INSERT DATA IN LINK TABLE SECTION FOR THAT NODE
ANY MORE RECORDS
YES
NO
5
5I

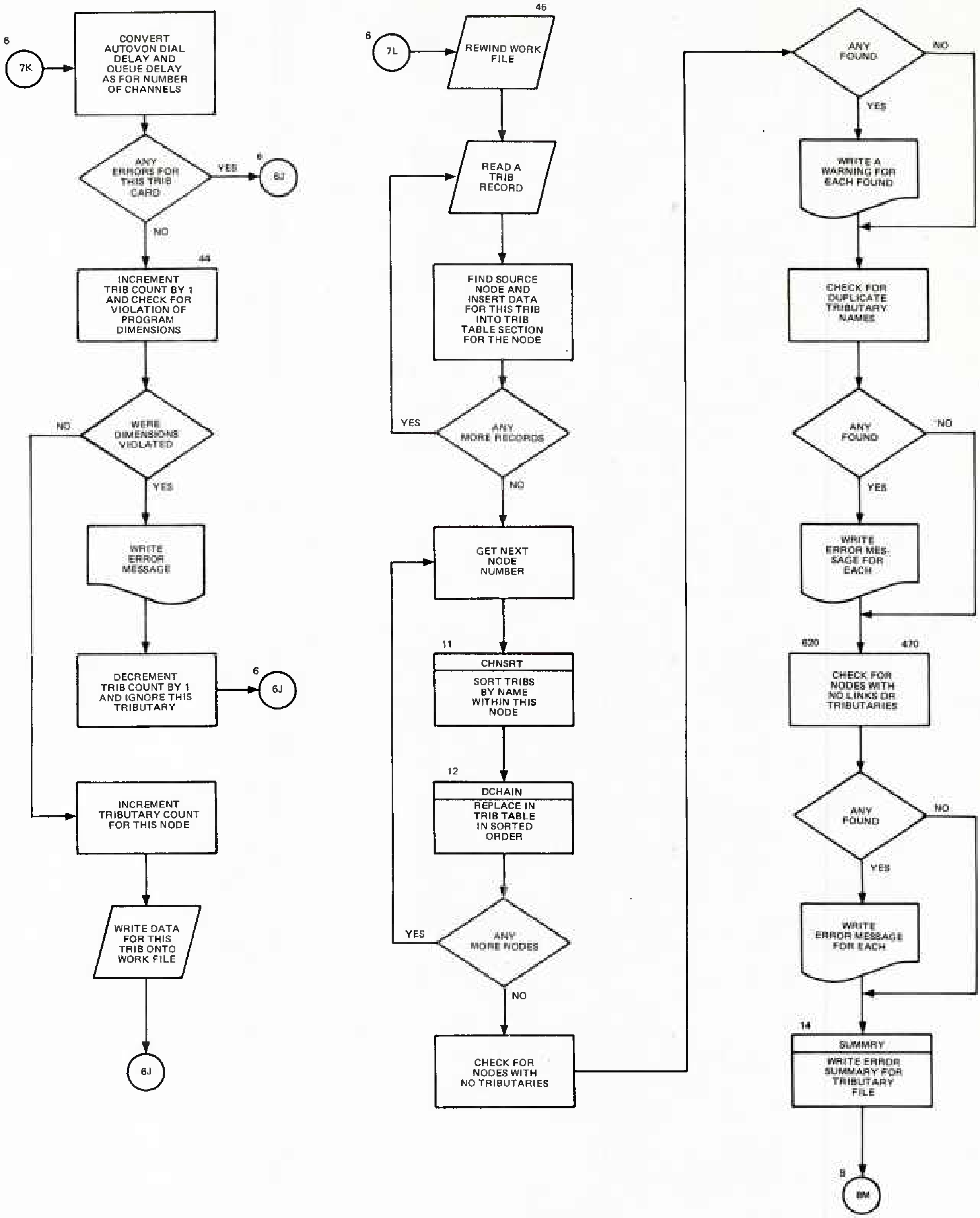
Status Generator Program (Sheet 4 of 14)


4
5I
GET NEXT NODE NUMBER
11
CHNSRT
SORT LINKS BY NAME WITHIN THIS NODE
12
DCHAIN
REPLACE IN TABLE IN SORTED ORDER
ANY MORE NODES
YES
NO
36
CHECK FOR NODES WHICH HAVE NO LINKS
ANY WITH NO LINKS
NO
YES
WRITE A WARNING FOR EACH
366
CHECK FOR NODES WITH LINKS TO ONLY ONE OTHER NODE
ANY FOUND
NO
YES
WRITE WARNING FOR EACH
CHECK FOR LINKS WITH MORE THAN 255 CHANNELS
ANY FOUND
NO
YES
WRITE ERROR MESSAGE FOR EACH
CHECK FOR LINK NAMES FROM ONE SOURCE NODE BUT TO MORE THAN ONE DESTINATION NODE
ANY FOUND
NO
YES
WRITE ERROR MESSAGE FOR EACH
CHECK FOR MULTIPLE INPUT FOR A GIVEN LINK
ANY FOUND
NO
YES
WRITE AN ERROR MESSAGE FOR EACH
CHECK FOR MULTIPLE LINK NAMES USED FOR SAME PAIR OF NODES
ANY FOUND
NO
YES
PRINT ERROR FOR EACH FOUND
CHECK FOR LINK NAMES BEING USED BETWEEN MORE THAN 1 PAIR OF DIFFERENT NODES
ANY FOUND
NO
YES
WRITE ERROR MESSAGE FOR EACH
CHECK FOR ALL LINK CARDS AT A LINK HAVING SAME SECURITY AND QUEUE DELAY
ANY FOUND
NO
YES
USE 1ST DEFINITION AND WRITE WARNING
14
391
SUMMRY
WRITE ERROR SUMMARY FOR LINK FILE
6
6J

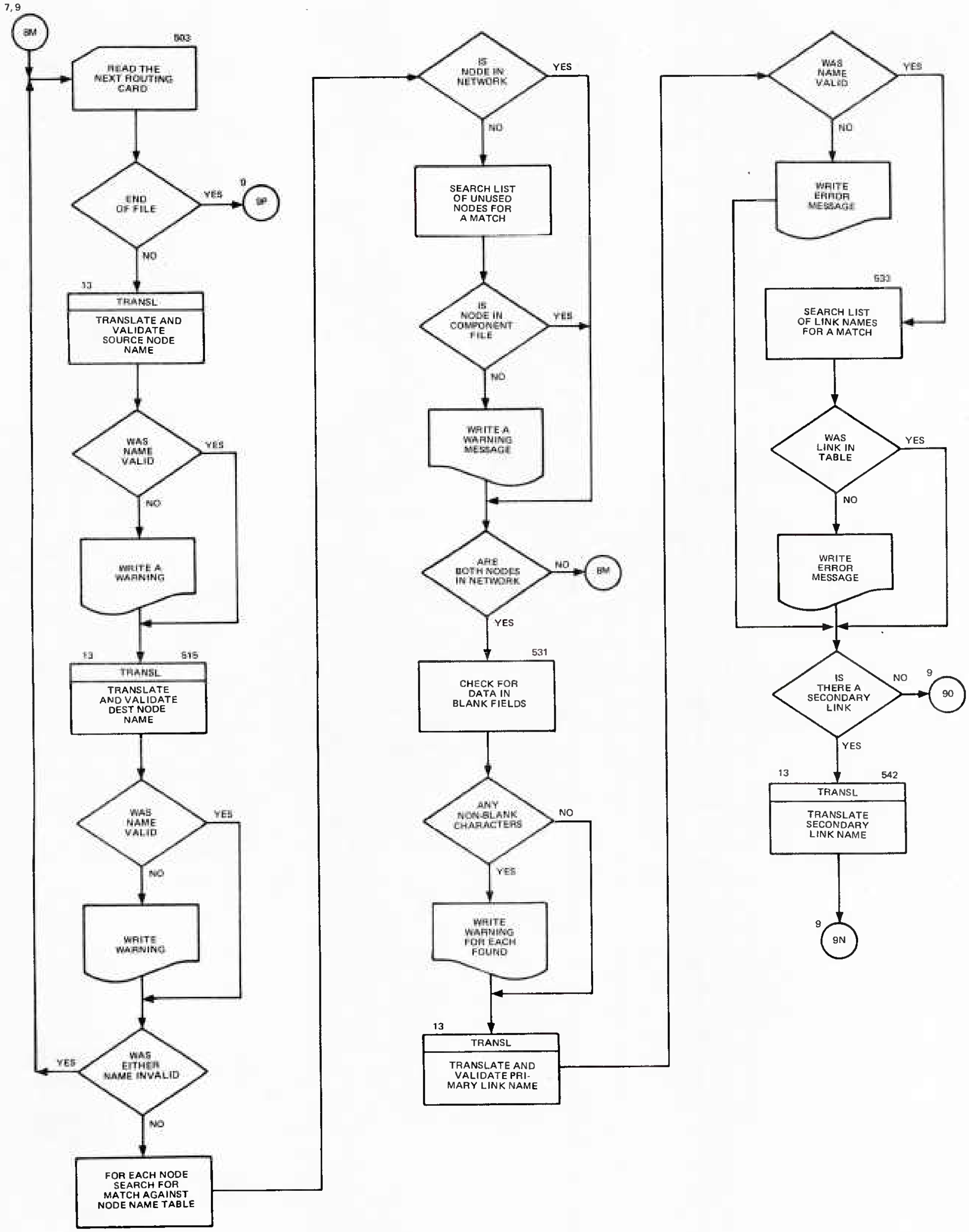
Status Generator Program (Sheet 5 of 14)


5, 7
6J
403
READ THE NEXT TRIB CARD
END OF FILE
YES
7
7L
NO
13
TRANSL
TRANSLATE AND VALIDATE NODE NAME
IS NAME VALID
YES
NO
WRITE A WARNING MESSAGE
410
SEARCH FOR MATCH AGAINST NODE NAME TABLE
IS NODE IN NETWORK
YES
NO
SEARCH UNUSED NODE NAMES FOR A MATCH
IS NODE IN COMPONENT FILE
YES
NO
WRITE WARNING MESSAGE
420
CHECK BLANK FIELDS FOR NON-BLANK CHARACTERS
ANY FOUND
NO
YES
WRITE WARNING MESSAGE FOR EACH
13
TRANSL
TRANSLATE AND VALIDATE TRIB NAME
IS NAME VALID
YES
NO
PRINT ERROR MESSAGE
14
422
CONVRT
CONVERT NUMBER OF CHANNELS TO AN INTEGER
ANY ERROR
YES
NO
IS VALUE IN RANGE
YES
NO
PRINT ERROR MESSAGE
423
TRANSLATE SECURITY INDICATOR BY
A → 1
T → 2
S → 3
L → 4
U → 5
WAS INDICATOR FOUND
YES
NO
PRINT ERROR MESSAGE
CONVERT TELETYPE, PUNCHED CARD, MAGNETIC TAPE AND END OF MESSAGE ACKNOWLEDGEMENT DELAYS AS FOR NUMBER OF CHANNELS
14
CONVRT
CONVERT LTC NUMBER
CONVERSION ERROR
YES
NO
IS VALUE LE LTC INPUT FOR THIS NODE
YES
NO
PRINT ERROR MESSAGE
7
7K

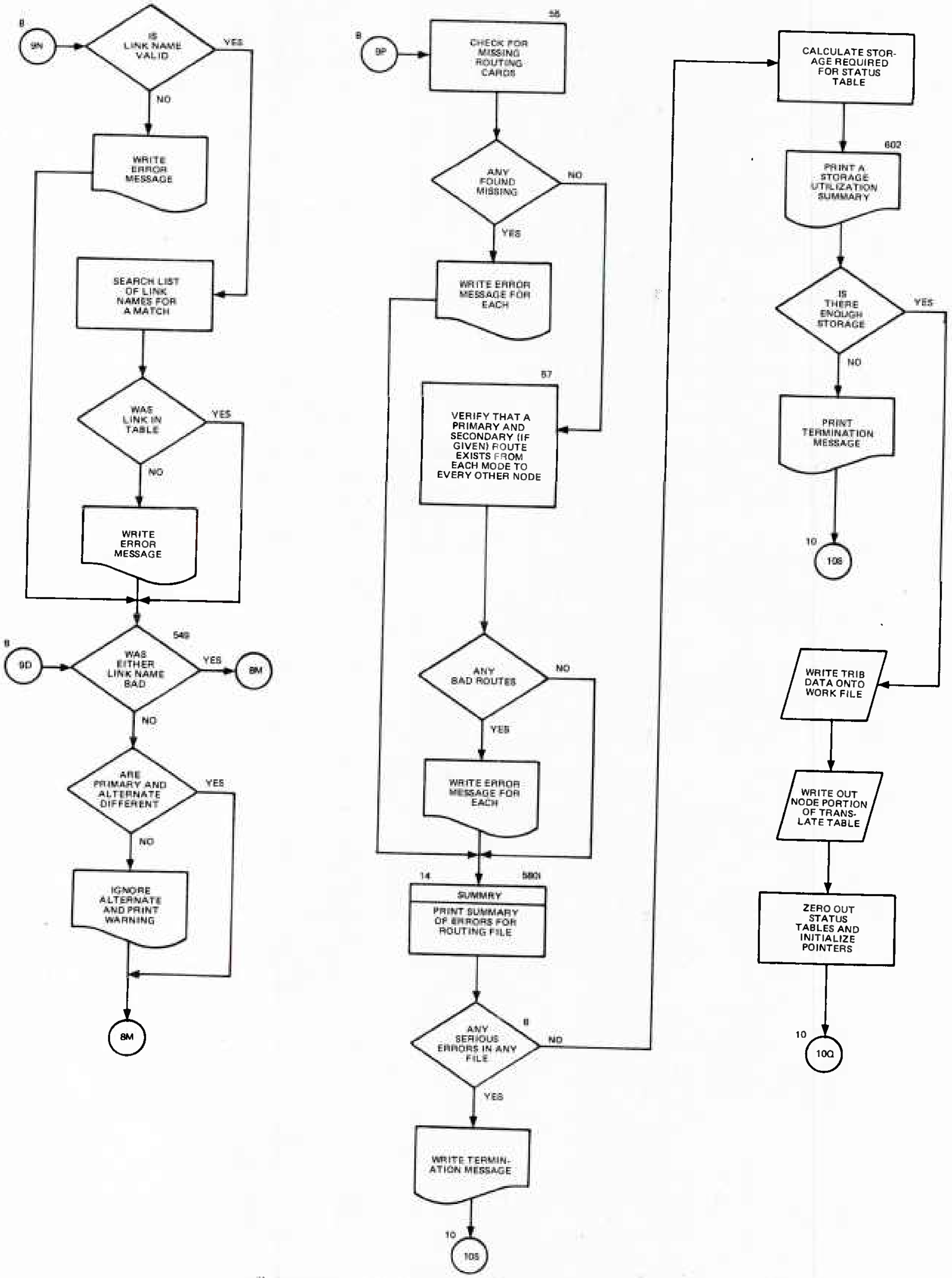
Status Generator Program (Sheet 6 of 14)


6
7K
CONVERT AUTOVON DIAL DELAY AND QUEUE DELAY AS FOR NUMBER OF CHANNELS
ANY ERRORS FOR THIS TRIB CARD
YES
6
6J
NO
44
INCREMENT TRIB COUNT BY 1 AND CHECK FOR VIOLATION OF PROGRAM DIMENSIONS
WERE DIMENSIONS VIOLATED
NO
YES
WRITE ERROR MESSAGE
DECREMENT TRIB COUNT BY 1 AND IGNORE THIS TRIBUTARY
6
6J
INCREMENT TRIBUTARY COUNT FOR THIS NODE
WRITE DATA FOR THIS TRIB ONTO WORK FILE
6J
6
7L
45
REWIND WORK FILE
READ A TRIB RECORD
FIND SOURCE NODE AND INSERT DATA FOR THIS TRIB INTO TRIB TABLE SECTION FOR THE NODE
ANY MORE RECORDS
YES
NO
GET NEXT NODE NUMBER
11
CHNSRT
SORT TRIBS BY NAME WITHIN THIS NODE
12
DCHAIN
REPLACE IN TRIB TABLE IN SORTED ORDER
ANY MORE NODES
YES
NO
CHECK FOR NODES WITH NO TRIBUTARIES
ANY FOUND
NO
YES
WRITE A WARNING FOR EACH FOUND
CHECK FOR DUPLICATE TRIBUTARY NAMES
ANY FOUND
NO
YES
WRITE ERROR MES-SAGE FOR EACH
620
470
CHECK FOR NODES WITH NO LINKS OR TRIBUTARIES
ANY FOUND
NO
YES
WRITE ERROR MESSAGE FOR EACH
14
SUMMRY
WRITE ERROR SUMMARY FOR TRIBUTARY FILE
8
8M

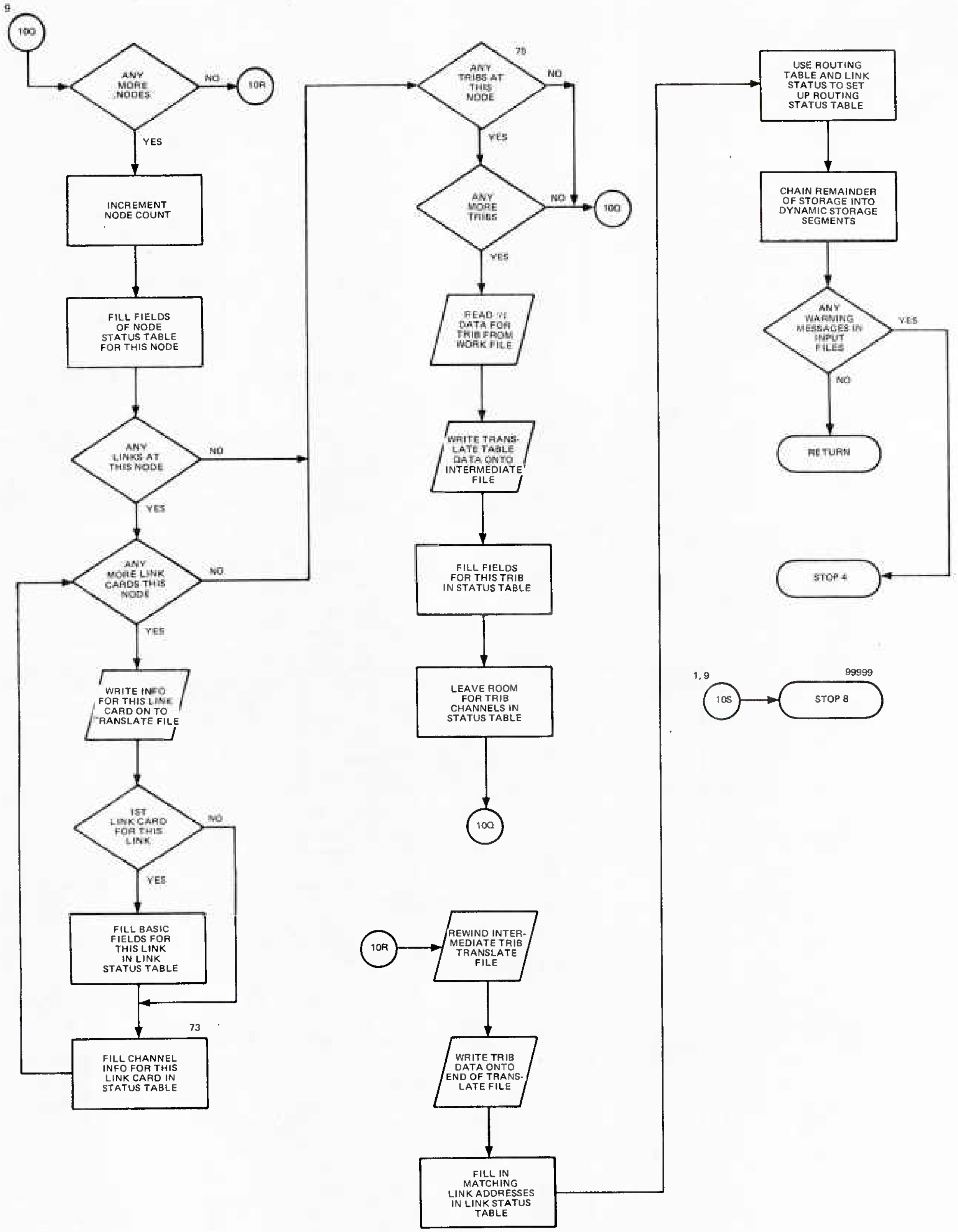
Status Generator Program (Sheet 7 of 14)


7, 9
8M
503
READ THE NEXT ROUTING CARD
END OF FILE
YES
9
9P
NO
13
TRANSL
TRANSLATE AND VALIDATE SOURCE NODE NAME
WAS NAME VALID
YES
NO
WRITE A WARNING
13
515
TRANSL
TRANSLATE AND VALIDATE DEST NODE NAME
WAS NAME VALID
YES
NO
WRITE WARNING
WAS EITHER NAME INVALID
YES
NO
FOR EACH NODE SEARCH FOR MATCH AGAINST NODE NAME TABLE
IS NODE IN NETWORK
YES
NO
SEARCH LIST OF UNUSED NODES FOR A MATCH
IS NODE IN COMPONENT FILE
YES
NO
WRITE A WARNING MESSAGE
ARE BOTH NODES IN NETWORK
NO
8M
YES
531
CHECK FOR DATA IN BLANK FIELDS
ANY NON-BLANK CHARACTERS
NO
YES
WRITE WARNING FOR EACH FOUND
13
TRANSL
TRANSLATE AND VALIDATE PRI-MARY LINK NAME
WAS NAME VALID
YES
NO
WRITE ERROR MESSAGE
533
SEARCH LIST OF LINK NAMES FOR A MATCH
WAS LINK IN TABLE
YES
NO
WRITE ERROR MESSAGE
IS THERE A SECONDARY LINK
NO
9
90
YES
13
542
TRANSL
TRANSLATE SECONDARY LINK NAME
9
9N

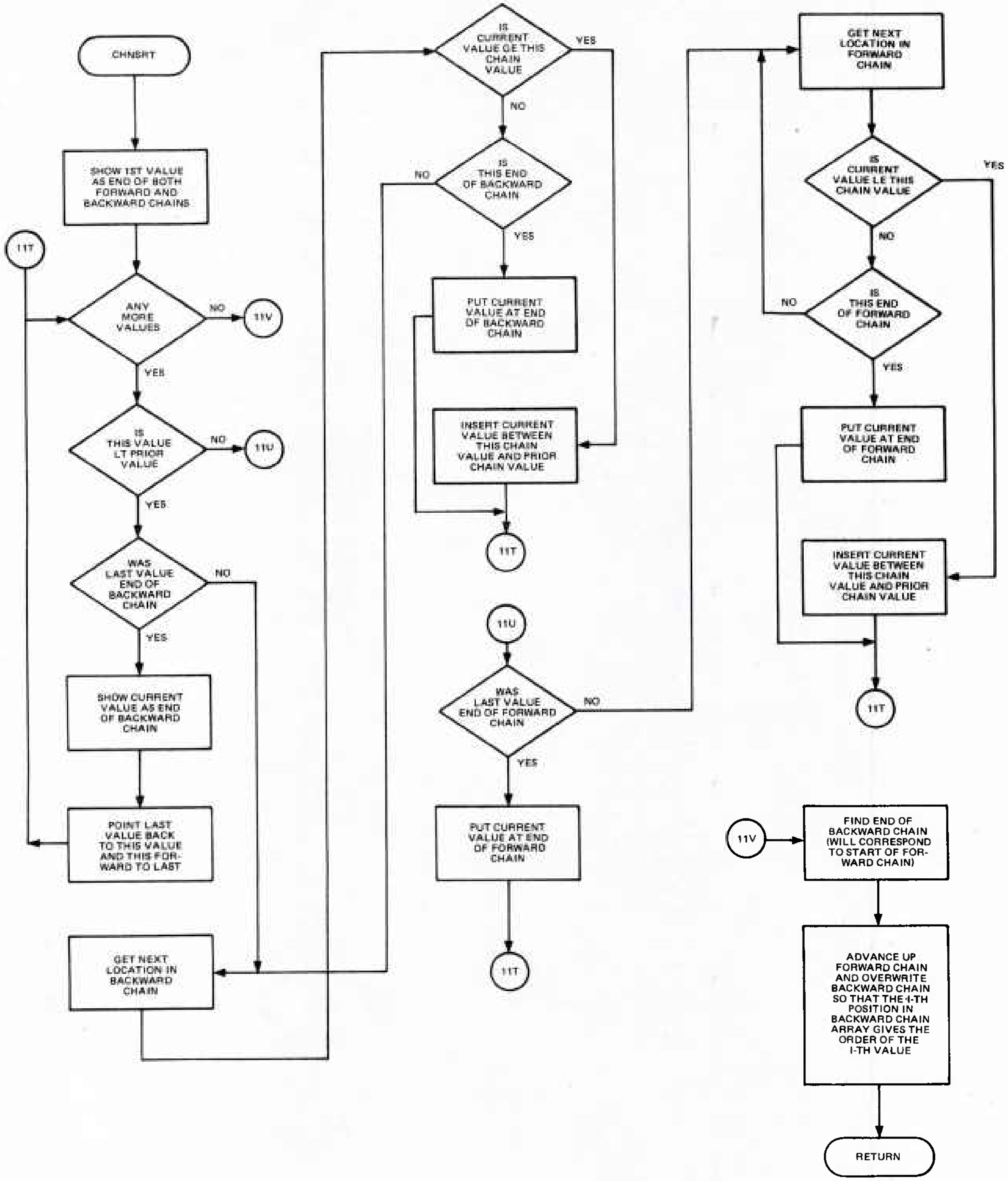
Status Generator Program (Sheet 8 of 14)


9
9N
IS LINK NAME VALID
YES
NO
WRITE ERROR MESSAGE
SEARCH LIST OF LINK NAMES FOR A MATCH
WAS LINK IN TABLE
YES
NO
WRITE ERROR MESSAGE
9
9D
549
WAS EITHER LINK NAME BAD
YES
8M
NO
ARE PRIMARY AND ALTERNATE DIFFERENT
YES
NO
IGNORE ALTERNATE AND PRINT WARNING
8M
9
9P
55
CHECK FOR MISSING ROUTING CARDS
ANY FOUND MISSING
NO
YES
WRITE ERROR MESSAGE FOR EACH
57
VERIFY THAT A PRIMARY AND SECONDARY (IF GIVEN) ROUTE EXISTS FROM EACH NODE TO EVERY OTHER NODE
ANY BAD ROUTES
NO
YES
WRITE ERROR MESSAGE FOR EACH
14
5801
SUMMARY
PRINT SUMMARY OF ERRORS FOR ROUTING FILE
8
ANY SERIOUS ERRORS IN ANY FILE
NO
YES
WRITE TERMINATION MESSAGE
10
10S
CALCULATE STORAGE REQUIRED FOR STATUS TABLE
602
PRINT A STORAGE UTILIZATION SUMMARY
IS THERE ENOUGH STORAGE
YES
NO
PRINT TERMINATION MESSAGE
10
10S
WRITE TRIB DATA ONTO WORK FILE
WRITE OUT NODE PORTION OF TRANSLATE TABLE
ZERO OUT STATUS TABLES AND INITIALIZE POINTERS
10
10Q

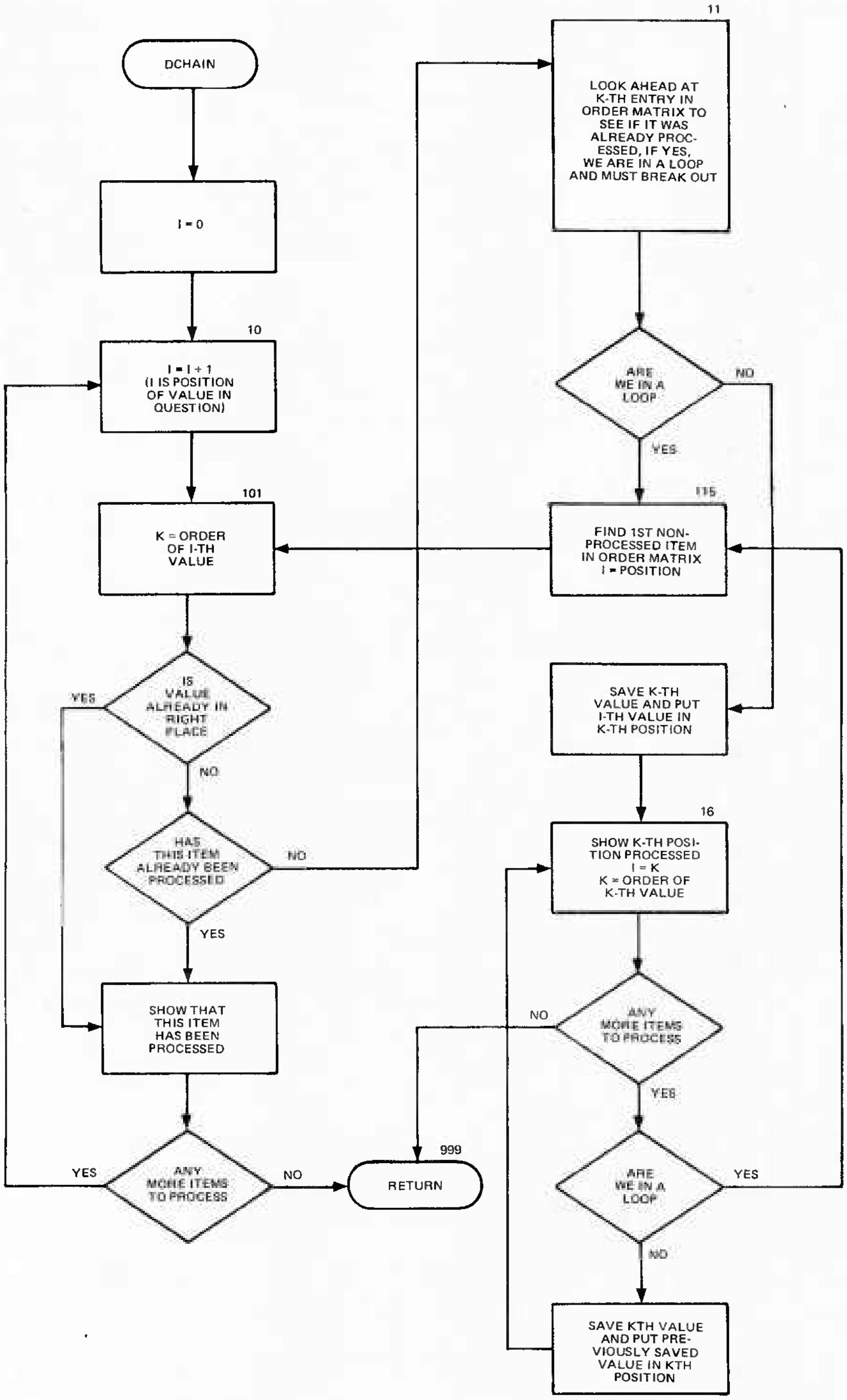
Status Generator Program (Sheet 9 of 14)


9
10Q
ANY MORE NODES
NO
10R
YES
INCREMENT NODE COUNT
FILL FIELDS OF NODE STATUS TABLE FOR THIS NODE
ANY LINKS AT THIS NODE
NO
YES
ANY MORE LINK CARDS THIS NODE
NO
YES
WRITE INFO FOR THIS LINK CARD ON TO TRANSLATE FILE
1ST LINK CARD FOR THIS LINK
NO
YES
FILL BASIC FIELDS FOR THIS LINK IN LINK STATUS TABLE
73
FILL CHANNEL INFO FOR THIS LINK CARD IN STATUS TABLE
75
ANY TRIBS AT THIS NODE
NO
YES
ANY MORE TRIBS
NO
10Q
YES
READ IN DATA FOR TRIB FROM WORK FILE
WRITE TRANSLATE TABLE DATA ONTO INTERMEDIATE FILE
FILL FIELDS FOR THIS TRIB IN STATUS TABLE
LEAVE ROOM FOR TRIB CHANNELS IN STATUS TABLE
10Q
10R
REWIND INTERMEDIATE TRIB TRANSLATE FILE
WRITE TRIB DATA ONTO END OF TRANSLATE FILE
FILL IN MATCHING LINK ADDRESSES IN LINK STATUS TABLE
USE ROUTING TABLE AND LINK STATUS TO SET UP ROUTING STATUS TABLE
CHAIN REMAINDER OF STORAGE INTO DYNAMIC STORAGE SEGMENTS
ANY WARNING MESSAGES IN INPUT FILES
YES
NO
RETURN
STOP 4
1, 9
99999
10S
STOP 8

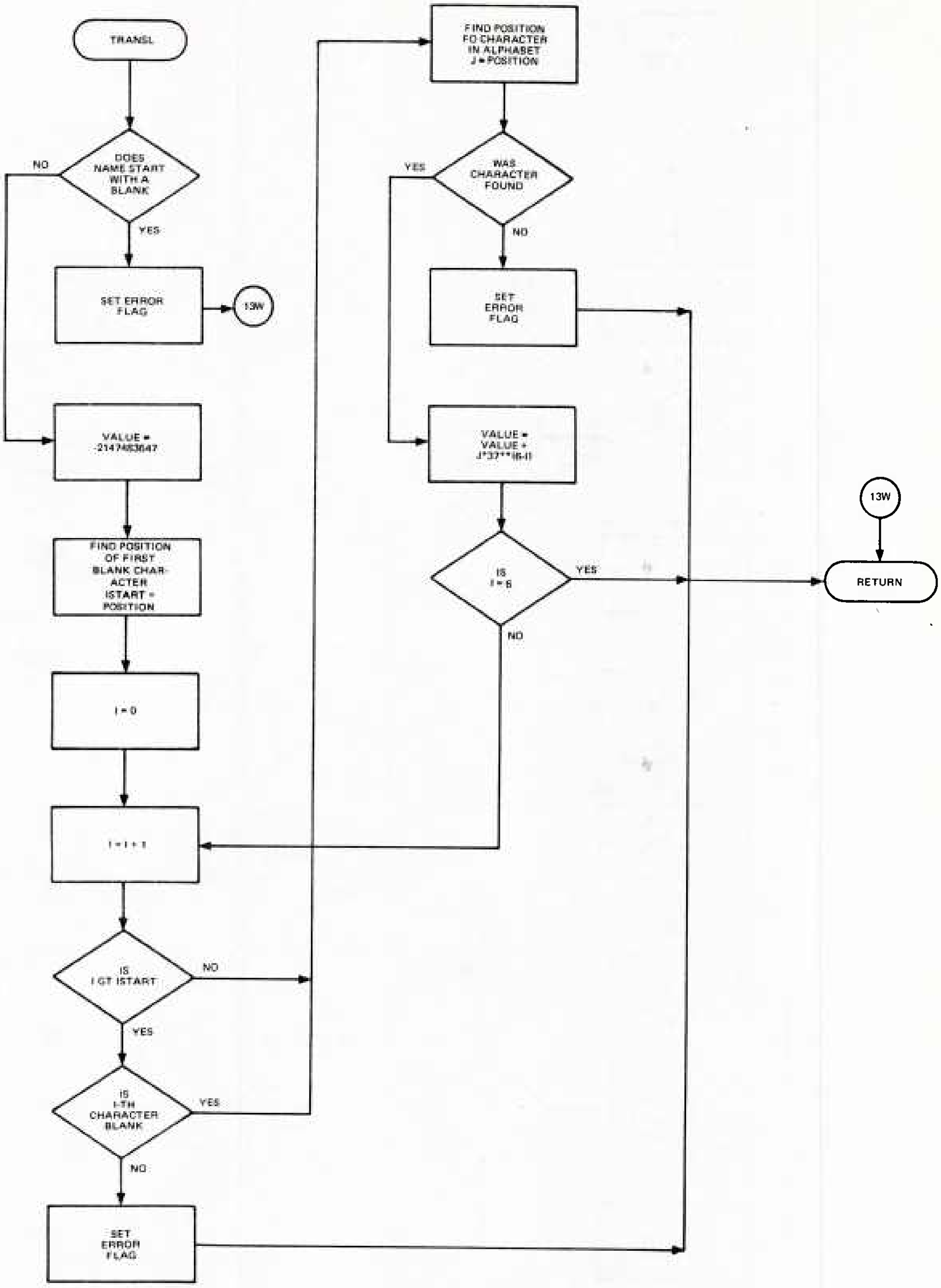
Status Generator Program (Sheet 10 of 14)


CHNSRT
SHOW 1ST VALUE AS END OF BOTH FORWARD AND BACKWARD CHAINS
11T
ANY MORE VALUES
NO
11V
YES
IS THIS VALUE LT PRIOR VALUE
NO
11U
YES
WAS LAST VALUE END OF BACKWARD CHAIN
NO
YES
SHOW CURRENT VALUE AS END OF BACKWARD CHAIN
POINT LAST VALUE BACK TO THIS VALUE AND THIS FOR-WARD TO LAST
GET NEXT LOCATION IN BACKWARD CHAIN
IS CURRENT VALUE GE THIS CHAIN VALUE
YES
NO
IS THIS END OF BACKWARD CHAIN
NO
YES
PUT CURRENT VALUE AT END OF BACKWARD CHAIN
INSERT CURRENT VALUE BETWEEN THIS CHAIN VALUE AND PRIOR CHAIN VALUE
11T
11U
WAS LAST VALUE END OF FORWARD CHAIN
NO
YES
PUT CURRENT VALUE AT END OF FORWARD CHAIN
11T
GET NEXT LOCATION IN FORWARD CHAIN
IS CURRENT VALUE LE THIS CHAIN VALUE
YES
NO
IS THIS END OF FORWARD CHAIN
NO
YES
PUT CURRENT VALUE AT END OF FORWARD CHAIN
INSERT CURRENT VALUE BETWEEN THIS CHAIN VALUE AND PRIOR CHAIN VALUE
11T
11V
FIND END OF BACKWARD CHAIN (WILL CORRESPOND TO START OF FOR-WARD CHAIN)
ADVANCE UP FORWARD CHAIN AND OVERWRITE BACKWARD CHAIN SO THAT THE I-TH POSITION IN BACKWARD CHAIN ARRAY GIVES THE ORDER OF THE I-TH VALUE
RETURN

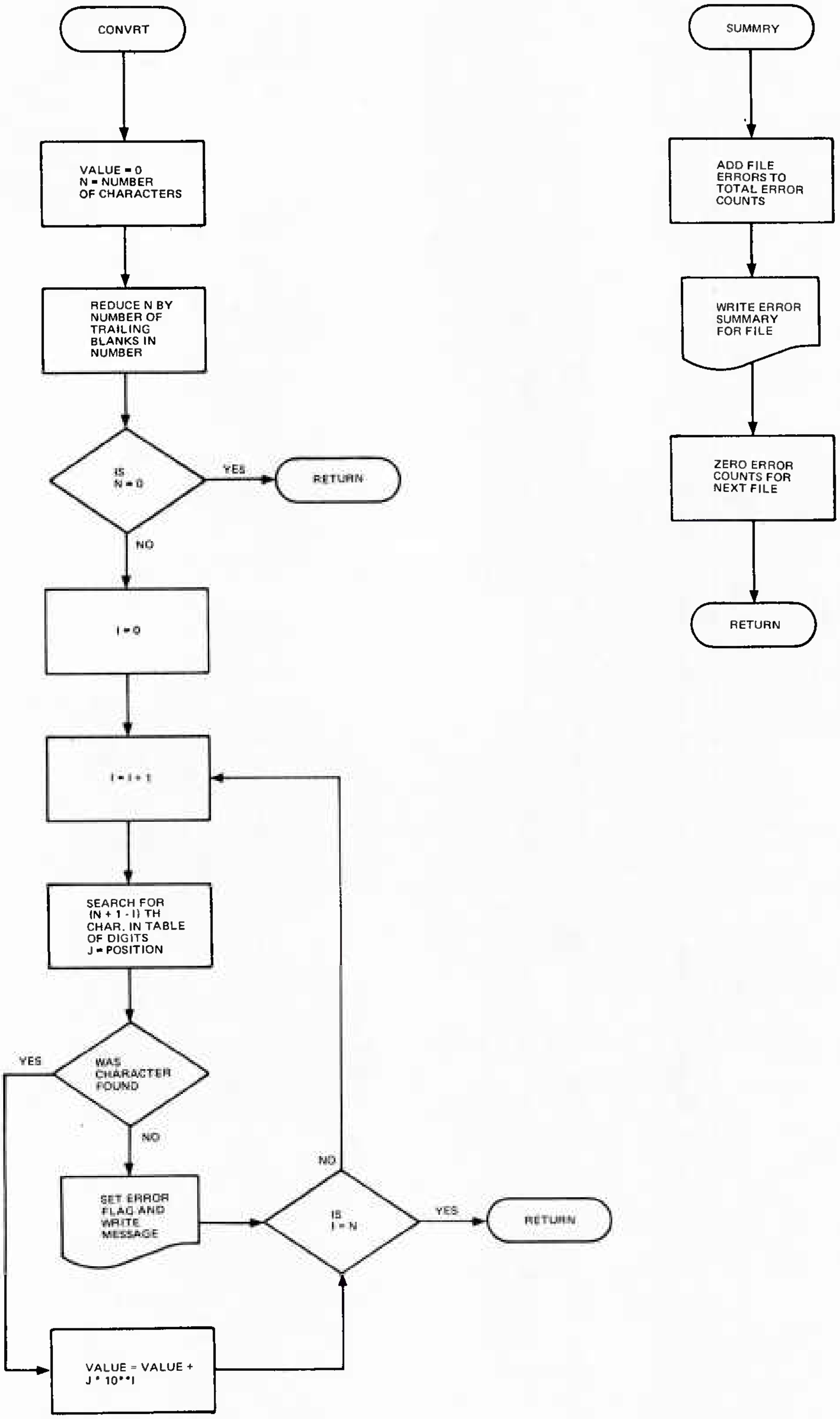
Status Generator Program (Sheet 11 of 14)



Status Generator Program (Sheet 12 of 14)



Status Generator Program (Sheet 13 of 14)


CONVRT
VALUE = 0
N = NUMBER OF CHARACTERS
REDUCE N BY NUMBER OF TRAILING BLANKS IN NUMBER
IS N = 0
YES
RETURN
NO
I = 0
I = I + 1
SEARCH FOR (N + 1 - I) TH CHAR. IN TABLE OF DIGITS J = POSITION
WAS CHARACTER FOUND
YES
NO
SET ERROR FLAG AND WRITE MESSAGE
NO
IS I = N
YES
RETURN
VALUE = VALUE + J * 10**I
SUMMRY
ADD FILE ERRORS TO TOTAL ERROR COUNTS
WRITE ERROR SUMMARY FOR FILE
ZERO ERROR COUNTS FOR NEXT FILE
RETURN

Status Generator Program (Sheet 14 of 14)

SECTION 3 - DISCRETE MESSAGE GENERATOR

3.1 PROGRAM REQUIREMENT DESCRIPTION

3.1.1 Program Name: Discrete Message Generator

3.1.2 Program Identification: DNDISMG1

3.1.3 Purpose

The Discrete Message Generator program creates a message file suitable for use by the Basic Simulation program from traffic data derived from the Header Extract Tapes provided by each of the automated AUTODIN Switching Centers.

3.1.4 Requirements

The program is written in American National Standards Institute (ANSI) COBOL to make the program machine-independent. Initial implementation is on an IBM Model 360/65 and requires approximately 175K bytes of memory for execution.

3.2 INPUT SPECIFICATIONS

Three input files are required by the Discrete Message Generator; the Sorted Traffic, Translation, and Control files.

3.2.1 Sorted Traffic

The Sorted Traffic file is created by the Assemble Traffic and the Traffic Modification and Sort programs. Its format is described in Table 11-2.

3.2.2 Translation File

The Translation file is created by the Status Generator program and is described in Tables 2-8 through 2-11.

3.2.3 Control File

The Control file consists of a Parameter card, followed by an optional Report Title card. The Parameter card is required and is described in Table 3-1.

The Report Title card is needed only if the Message List or Summary/Overflow option is specified on the Parameter card. The data on the card is printed as a report title at the top of each report page. If the card is omitted, no report title is generated. If no reports are requested and the Report Title card is present, it is ignored.

3.3 OUTPUT SPECIFICATION

The Discrete Message Generator program output is the Message file which contains the messages generated by the program. The output message format is described in Tables 3-2 and 3-3. Messages which have more than five destinations are described by two or more output records (in card-image form). The first card contains up to five destinations, and each successive card can have up to nine destinations. The additional message continuation cards use a different format than the first card, however the same data fields are used. The cards for a given message are sequenced in Columns 1 and 2.

3.4 PROGRAM LOGIC

The program input consists of a file of partial messages which describe source-to-destination tributary pairs. A full message consists of one or more of these elementary descriptors. Each descriptor is time-stamped with its time-of-transmission (TOT) and also contains the message's serial number, precedence, security level, length, and format. The input file created by the Assemble Traffic program is sorted by the Traffic Modification and Sort program by origin tributary, TOT, serial number, security, and precedence. The Discrete Message Generator assembles whole messages from the individual pieces and reformats the information in a form acceptable to Basic Simulation. Depending on the program input parameters, data can be filtered by time, and various reports can be generated.

3.4.1 Program Initialization

Program execution begins with reading in the Parameter card. Where possible, minor errors are ignored and default values substituted. If any uncorrectable errors occurred, the program is terminated after printing appropriate diagnostic messages.

Next, the Translation file is read into core. The component or switch names are taken from the Component cards, and the tributary names, number of channels, and transmission delay times (teletype, card, mag-tape) are extracted from the Tributary cards. No link information is used. Any error in reading in this file causes program termination.

The six-character tributary names are presumed to be structured such that the first three characters identify the switch and the last three characters identify the tributary channel at that switch. A further assumption is that there will be no more than 256 tributaries at any one switch. A major portion of program processing time is used to identify the source and destination in each input record. To facilitate this, a binary search capability on the tributary name is necessary. This is accomplished through the use of table TRIBUTARY-NAME-TABLE. This table is structured such that each possible switch in the network (to a maximum of 31) is allocated 256 3-byte locations in the table. The switches in the Translate file are in sorted order and the tributaries within each switch are also in order. As each tributary is read in, the 3-byte tributary identification is stripped off and inserted in the TRIBUTARY-NAME-TABLE in the next available location in the section corresponding to its switch. Empty locations in the table are blank-filled.

The table is somewhat wasteful of core as there is considerable unused space in it. Accordingly, the data items for the tributaries are put into their own tables sequentially as they are encountered in the Tributary file. These tables are:

NUMBER-OF-CHANNELS-TABLE
TELETYPE-SPEED-TABLE
CARD-SPEED-TABLE
TAPE-SPEED-TABLE

Their names are self-explanatory. This means that the tributary names and data items are not in corresponding locations in their respective tables. An additional table which relates the two is required. The TRIBUTARY-LOCATION-TABLE is used to provide this communication. It is structured in the same manner as TRIBUTARY-NAME-TABLE. As each tributary name is put into TRIBUTARY-NAME-TABLE, the position of its data in the four data tables is put into the corresponding location of TRIBUTARY-LOCATION-TABLE. Hence, whenever a search for a tributary name is made in TRIBUTARY-NAME-TABLE, the data for the tributary can be readily found.

3.4.2 Message Generation

Message generation involves collecting the individual source-destination pairs from each input record, assembling them into message descriptors containing a single source and from one to 255 destinations, and formatting them in a form acceptable to Basic Simulation.

The input records are sorted prior to processing by the Discrete Message Generator in a manner which guarantees that all of the records comprising a single message appear consecutively in the file. The primary sort field is by origin tributary, which allows summary statistics by tributary to be easily gathered. Further sort fields are TOT serial number, security, and precedence.

The gathering process begins with the first record on the data file. A binary search is made in the TRIBUTARY-NAME-TABLE for both source and destination tributaries. Once verified, the source, TOT, line block count, serial number, precedence, security, and language format are saved. The destination is saved in DESTINATIONS (1) as the first destination for the message.

Successive records are then read in from the Sorted Traffic file and checked for being in the same message. If the origin, TOT, serial number, precedence, security, and language format are all identical to the first record of the message, the record is a part of a multiple-destination message. It it is, its destination is added into DESTINATIONS.

If the preceding parameters are not all identical, the new record marks the beginning of the next message. If 255 destinations have been collected for a single message, the 256th destination is treated as the start of a new message.

At this point the current message must be handled. It is assigned the next sequential message number (the starting number is an input parameter). The TOT is computed as the greater of the input TOT or the time when the next channel at the source tributary will become free. An internal clock is maintained which is used to keep track of channel availability times. As each message is generated, a channel is selected for message transmission. The channel's next availability time is then computed as the current clock time plus message transmission time, plus intermessage delay, when specified. Precedence and security are converted to numeric values and the message length is entered as line blocks multiplied by 10.

The message is now written onto the output message file (FD DATA-OUT) in 80-character records, as described in Paragraph 3.3. If the Message List option is on and the TOT is within the range specified on the Parameter card, the message is reformatted for better visual comprehension and printed.

If the current input record has a source tributary which differs from the message just processed, all messages from this tributary have been processed and the program prints a summary report for this tributary. During message generation, various statistics have been collected on such items as number of messages for each data type, security, priority, and destination tributary. Depending on the Summary/Overflow option on the Parameter card, either a full summary report, a partial report showing messages overflowed (Paragraph 3.4.3), or a one-line summary per tributary is printed.

The current input record now becomes the first record in the next message. Its origin, message number, et cetera are saved, and the remainder of the message is gathered. Message generation continues until end of file is encountered in the input data set.

3.4.3 Time Filtering and Overflow

The input data file normally contains records describing an entire day's traffic flow through the AUTODIN network. It is possible to specify that traffic be generated for a given subset of the day. As each record is read into core, its TOT is checked to be sure it is within the Start and Stop Generation Times specified on the Parameter card. If it is, the record is processed as described in the previous paragraph, otherwise it is ignored.

As the TOT is calculated for each output message being generated, this time must also be checked for being less than the Stop Generation Time. If it is not, this is an overflow message. In this event, the overflow count for the origin tributary is incremented by one and the message is not written onto the output file.

3.4.4 Tributary Identification

As each record is processed, a check must be made as to whether its destination (or origin if it is the first record in a message) is included in the network as defined in the Translation file.

During program initialization, the three-character tributary identifiers for all valid tributaries in the network were placed into TRIBUTARY-NAME-TABLE. In addition, the unique three-character switch identifiers were placed in sorted order in NODE-NAME-TABLE.

To identify a tributary as being valid, its three-character switch identifier is extracted and a binary search made in NODE-NAME-TABLE for a match (the table is 31 locations long). Once the switch is identified, a further binary search based on the tributary identifier is made in the 256 location portion of TRIBUTARY-NAME-TABLE corresponding to the switch.

If the tributary is not located, special action must be taken, depending on whether this is an origin or destination tributary. In either event, all unique invalid names are collected in BAD-NAMES. If this is a destination, no further action is taken and the destination is inserted into the output message. If an origin, a flag is set in BAD-FLAG in the location corresponding to the name itself in BAD-NAMES. In addition, the input record is dropped. This action is required because the number of channels and transmission speeds for the tributary are not available.

When the last message has been processed by the program, the names and flags are printed out so that corrective action can be taken.

Program logic flow charts are included at the end of this section.

3.5 MACHINE DEPENDENCE

To the greatest extent possible, the Discrete Message Generator was made machine independent. All coding conforms to ANSI standards. However, in implementing the program on an IBM 360/65 some machine dependence was unavoidable. The Identification division describes an IBM 360/65. In addition, a number of computational variables were used in the program. Their PICTURE specifications are 9(9), which defines full-words on the 360/65.

Table 3-1. Discrete Message Generator Parameter Card

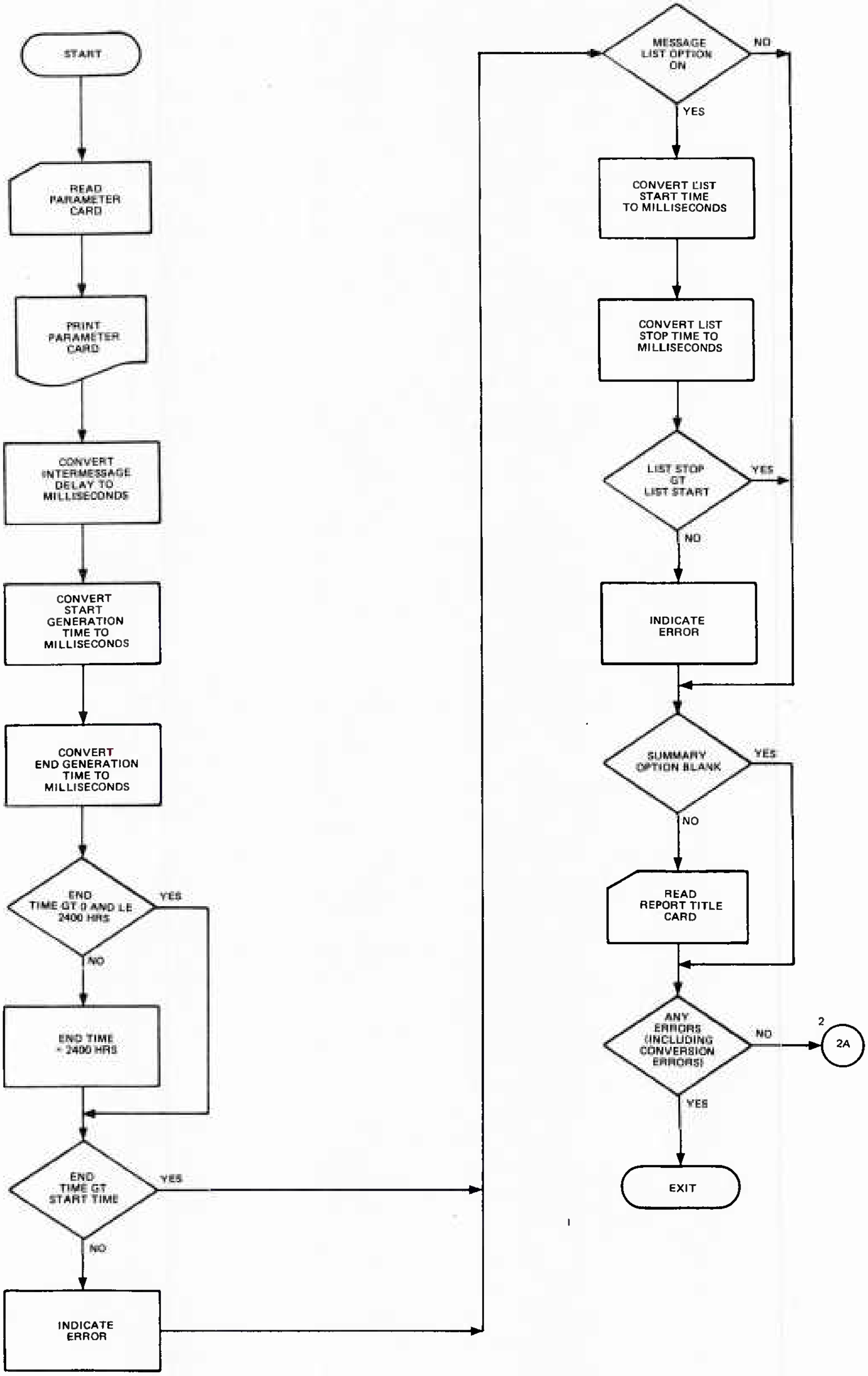
CARD COLMUN	DESCRIPTION	EXPLANATION
1-9	Inter Message Delay	Delay between transmissions, in tenths of seconds (Right-justified).
10-17	Start Message Number	May contain a start message number. If it is left blank, the start message number will default to 1. Must be right-justified.
18-21	Blank	
22-25	Start Message Generation Time	Start time for message generation and is a four-digit number containing the starting hour and minute in military time; e. g. , 1315.
26-29	Blank	
30-33	Stop Message Generation Time	Stop time for message generation and is a four-digit number containing the last hour and minute messages will be generated; e. g. , 1515.
34	Blank	
35-41	Message List Option	Must contain MSGLIST if a listing of messages as generated is desired. If blank, a message list is not generated.
42-45	Blank	
46-49	Message List Start Time	Start hour and minute for message listing.
50-53	Blank	
54-57	Message List Stop Time	Stop hour and minute for message listing.
58-64	Summary Option	May contain the characters SUMMARY, OVERFLO, or may be blank. If SUMMARY option is requested, a full summary listing by originating tributary will be generated. If OVERFLO option is requested, a summary listing of originating tributaries which generated overflow messages will be printed. If blank, a one line per tributary report will be generated.
65-67	RADAY	RADAY on which input data was extracted from the network.
68-80		Unused.

Table 3-2. Message Output Format

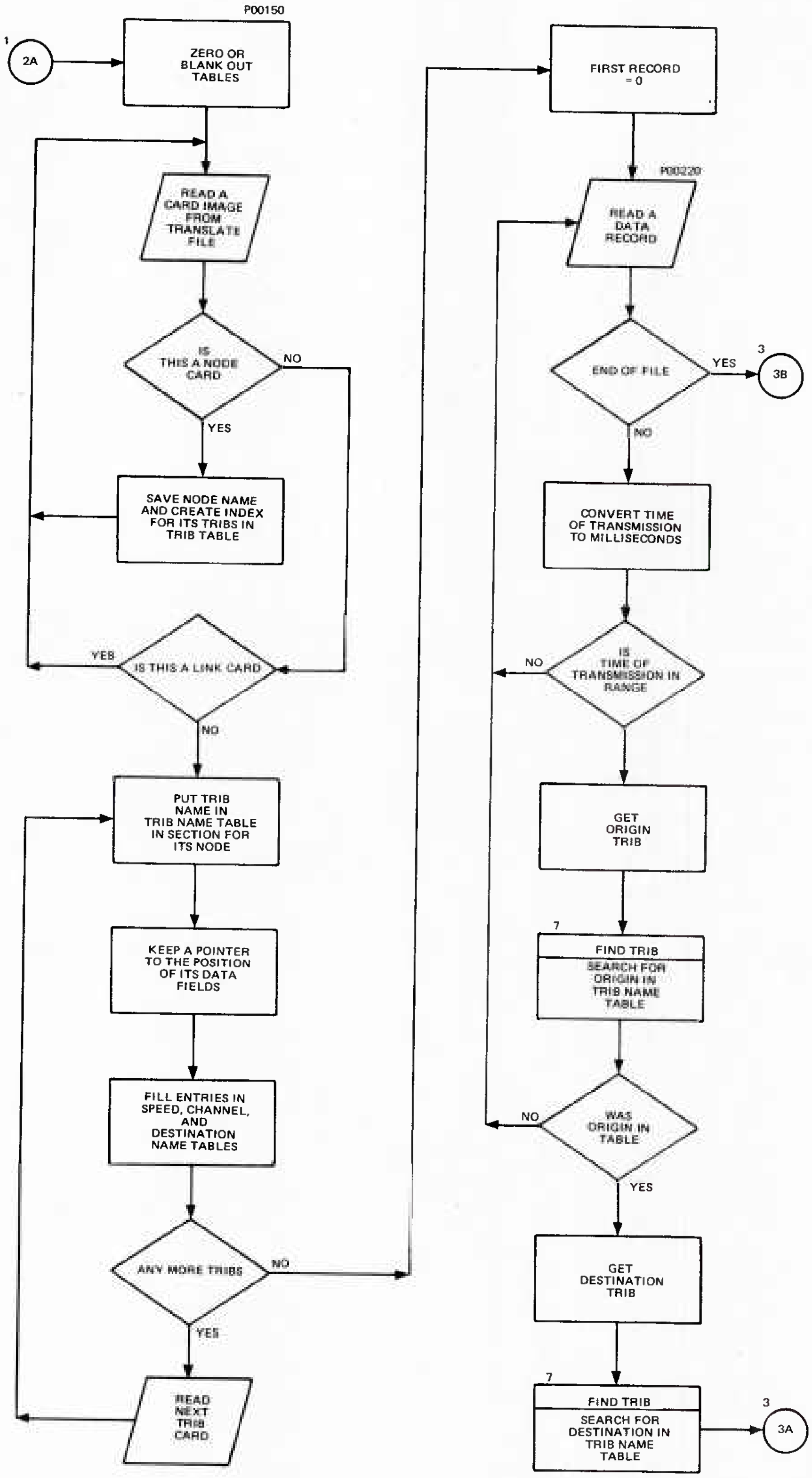
CARD COLUMN	DESCRIPTION	EXPLANATION
1-2	Card Sequence Number	01 for first card
3-11	Time	Time in milliseconds
12-13	Type	01 for Message Input
14	Blank	
15-20	Field 1	Message Number
21-26	Field 2	Originator
27-32	Field 3	Length
33-38	Field 4	Message Type
39-44	Field 5	Precedence
45-50	Field 6	Security
51-56	Field 7	1st Destination
57-62	Field 8	2nd Destination
63-68	Field 9	3rd Destination
69-74	Field 10	4th Destination
75-80	Field 11	5th Destination

Table 3-3. Message Continuation Card Format

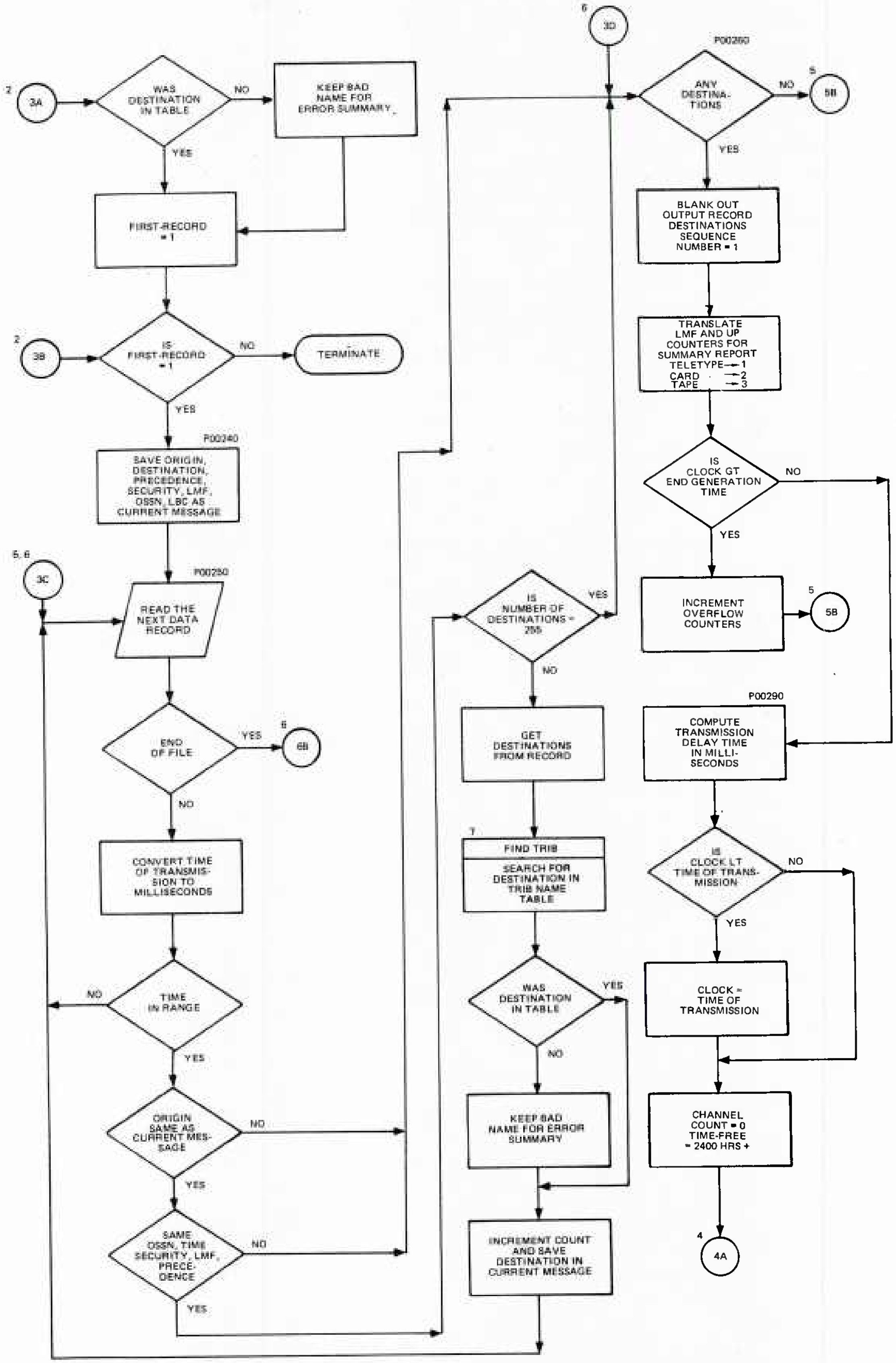
CARD COLUMN	DESCRIPTION	EXPLANATION
1-2	Card Sequence Number	02 or greater
3-11	Time	Time in milliseconds
12-13	Type	01 for Message Input
14	Blank	
15-20	Field 1	Message Number
21-26	Field 2	Originator
27-32	Field 3	6th, 15th, etc. Destination
33-38	Field 4	7th, 16th, etc.
39-44	Field 5	8th, 17th, etc.
45-50	Field 6	9th, 18th, etc.
51-56	Field 7	10, 19th, etc.
57-62	Field 8	11th, 20th, etc.
63-68	Field 9	12th, 21st, etc.
69-74	Field 10	13th, 22nd, etc.
75-80	Field 11	14th, 23rd, etc.


START
READ PARAMETER CARD
PRINT PARAMETER CARD
CONVERT INTERMESSAGE DELAY TO MILLISECONDS
CONVERT START GENERATION TIME TO MILLISECONDS
CONVERT END GENERATION TIME TO MILLISECONDS
END TIME GT 0 AND LE 2400 HRS
YES
NO
END TIME = 2400 HRS
END TIME GT START TIME
YES
NO
INDICATE ERROR
MESSAGE LIST OPTION ON
NO
YES
CONVERT LIST START TIME TO MILLISECONDS
CONVERT LIST STOP TIME TO MILLISECONDS
LIST STOP GT LIST START
YES
NO
INDICATE ERROR
SUMMARY OPTION BLANK
YES
NO
READ REPORT TITLE CARD
ANY ERRORS (INCLUDING CONVERSION ERRORS)
NO
2
2A
YES
EXIT

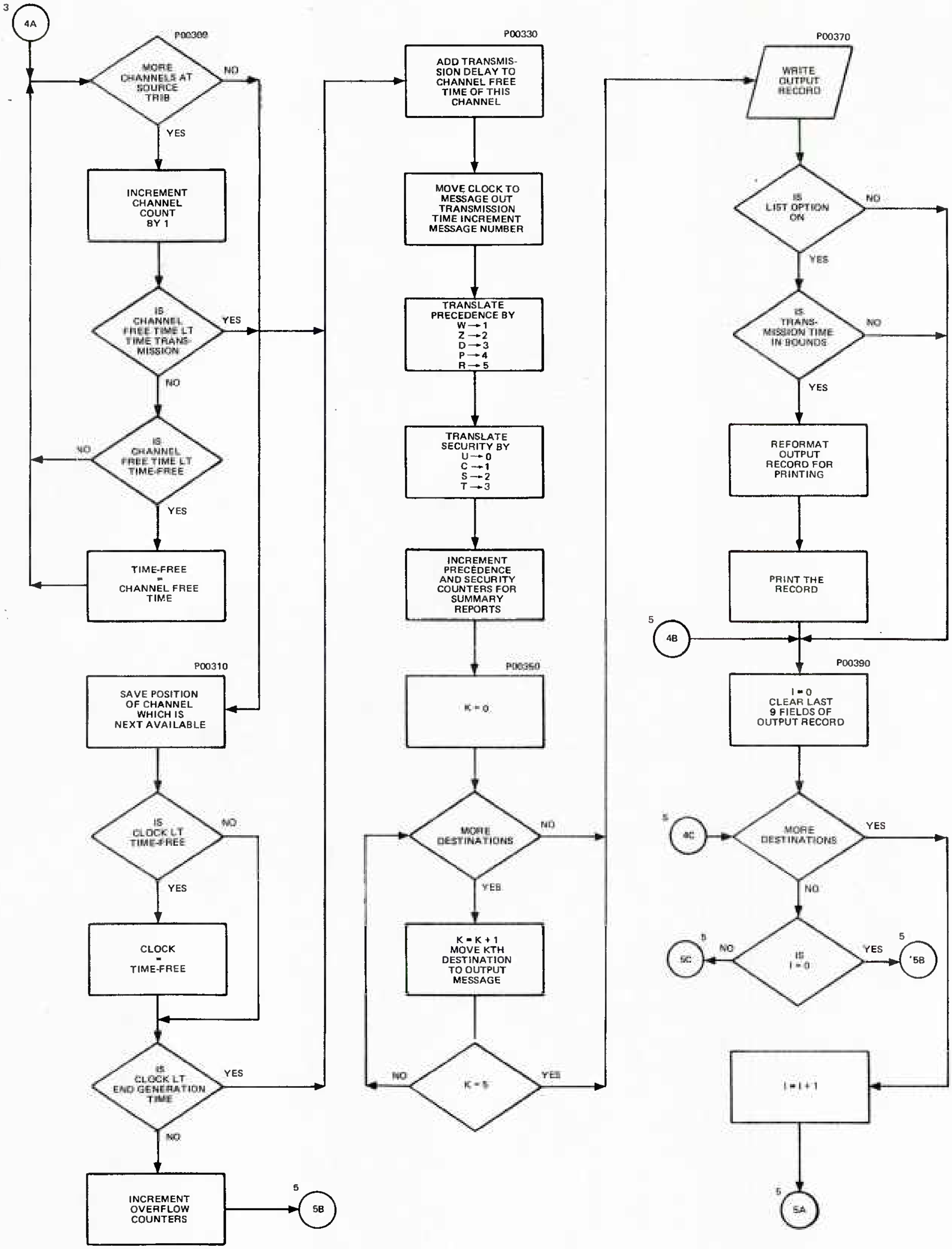
Discrete Message Generator Program (Sheet 1 of 7)


P00150
1
2A
ZERO OR BLANK OUT TABLES
READ A CARD IMAGE FROM TRANSLATE FILE
IS THIS A NODE CARD
NO
YES
SAVE NODE NAME AND CREATE INDEX FOR ITS TRIBS IN TRIB TABLE
YES
IS THIS A LINK CARD
NO
PUT TRIB NAME IN TRIB NAME TABLE IN SECTION FOR ITS NODE
KEEP A POINTER TO THE POSITION OF ITS DATA FIELDS
FILL ENTRIES IN SPEED, CHANNEL, AND DESTINATION NAME TABLES
ANY MORE TRIBS
NO
YES
READ NEXT TRIB CARD
FIRST RECORD = 0
P00220
READ A DATA RECORD
END OF FILE
YES
3
3B
NO
CONVERT TIME OF TRANSMISSION TO MILLISECONDS
NO
IS TIME OF TRANSMISSION IN RANGE
GET ORIGIN TRIB
7
FIND TRIB
SEARCH FOR ORIGIN IN TRIB NAME TABLE
NO
WAS ORIGIN IN TABLE
YES
GET DESTINATION TRIB
7
FIND TRIB
SEARCH FOR DESTINATION IN TRIB NAME TABLE
3
3A

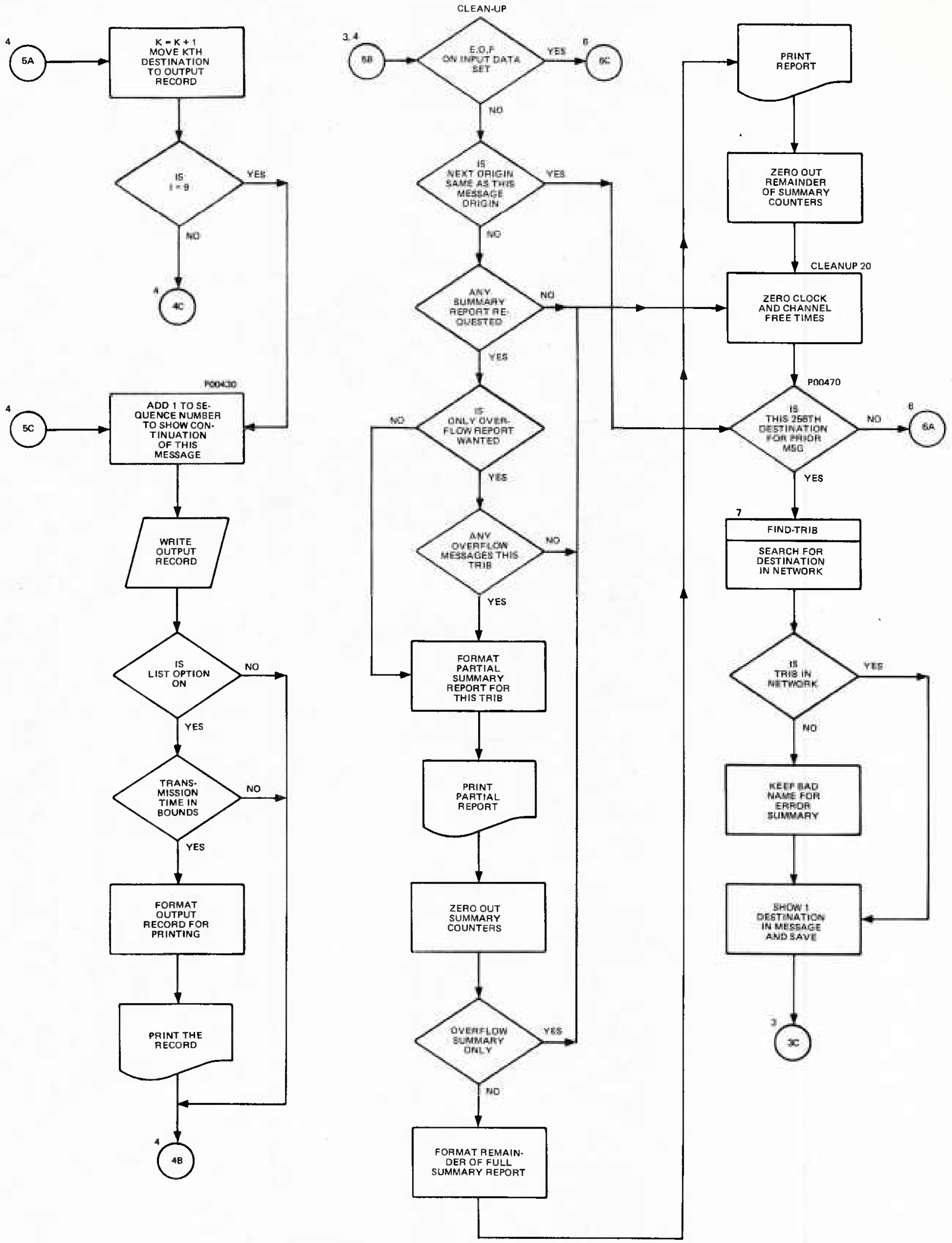
Discrete Message Generator Program (Sheet 2 of 7)


2
3A
WAS DESTINATION IN TABLE
NO
KEEP BAD NAME FOR ERROR SUMMARY
YES
FIRST-RECORD = 1
2
3B
IS FIRST-RECORD = 1
NO
TERMINATE
YES
P00240
SAVE ORIGIN, DESTINATION, PRECEDENCE, SECURITY, LMF, OSSN, LBC AS CURRENT MESSAGE
5, 6
3C
P00250
READ THE NEXT DATA RECORD
END OF FILE
YES
6
6B
NO
CONVERT TIME OF TRANSMISSION TO MILLISECONDS
TIME IN RANGE
NO
YES
ORIGIN SAME AS CURRENT MESSAGE
NO
YES
SAME OSSN, TIME SECURITY, LMF, PRECEDENCE
NO
YES
IS NUMBER OF DESTINATIONS = 255
YES
NO
GET DESTINATIONS FROM RECORD
7
FIND TRIB
SEARCH FOR DESTINATION IN TRIB NAME TABLE
WAS DESTINATION IN TABLE
YES
NO
KEEP BAD NAME FOR ERROR SUMMARY
INCREMENT COUNT AND SAVE DESTINATION IN CURRENT MESSAGE
6
3D
P00260
ANY DESTINATIONS
NO
5
5B
YES
BLANK OUT OUTPUT RECORD DESTINATIONS SEQUENCE NUMBER = 1
TRANSLATE LMF AND UP COUNTERS FOR SUMMARY REPORT
TELETYPE → 1
CARD → 2
TAPE → 3
IS CLOCK GT END GENERATION TIME
NO
YES
INCREMENT OVERFLOW COUNTERS
5
5B
P00290
COMPUTE TRANSMISSION DELAY TIME IN MILLISECONDS
IS CLOCK LT TIME OF TRANSMISSION
NO
YES
CLOCK = TIME OF TRANSMISSION
CHANNEL COUNT = 0
TIME-FREE = 2400 HRS +
4
4A

Discrete Message Generator Program (Sheet 3 of 7)

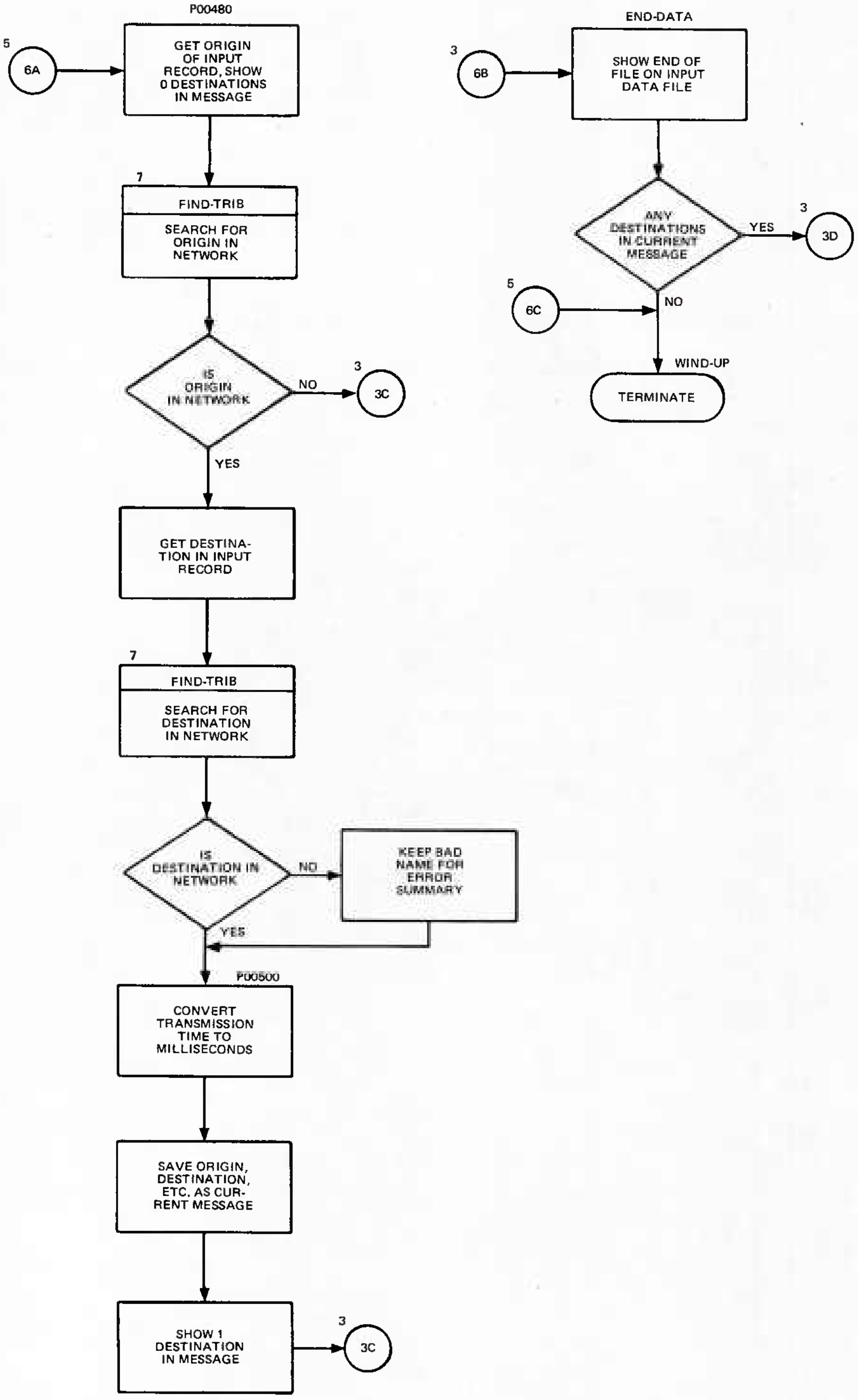

3
4A
P00309
MORE CHANNELS AT SOURCE TRIB
NO
YES
INCREMENT CHANNEL COUNT BY 1
IS CHANNEL FREE TIME LT TIME TRANS-MISSION
YES
NO
IS CHANNEL FREE TIME LT TIME-FREE
NO
YES
TIME-FREE = CHANNEL FREE TIME
P00310
SAVE POSITION OF CHANNEL WHICH IS NEXT AVAILABLE
IS CLOCK LT TIME-FREE
NO
YES
CLOCK = TIME-FREE
IS CLOCK LT END GENERATION TIME
YES
NO
INCREMENT OVERFLOW COUNTERS
5
5B
P00330
ADD TRANSMIS-SION DELAY TO CHANNEL FREE TIME OF THIS CHANNEL
MOVE CLOCK TO MESSAGE OUT TRANSMISSION TIME INCREMENT MESSAGE NUMBER
TRANSLATE PRECEDENCE BY W → 1 Z → 2 D → 3 P → 4 R → 5
TRANSLATE SECURITY BY U → 0 C → 1 S → 2 T → 3
INCREMENT PRECEDENCE AND SECURITY COUNTERS FOR SUMMARY REPORTS
P00350
K = 0
MORE DESTINATIONS
NO
YES
K = K + 1 MOVE KTH DESTINATION TO OUTPUT MESSAGE
K = 5
NO
YES
P00370
WRITE OUTPUT RECORD
IS LIST OPTION ON
NO
YES
IS TRANS-MISSION TIME IN BOUNDS
NO
YES
REFORMAT OUTPUT RECORD FOR PRINTING
PRINT THE RECORD
5
4B
P00390
I = 0 CLEAR LAST 9 FIELDS OF OUTPUT RECORD
5
4C
MORE DESTINATIONS
YES
NO
IS I = 0
NO
YES
5
5C
5
5B
I = I + 1
5
5A

Discrete Message Generator Program (Sheet 4 of 7)

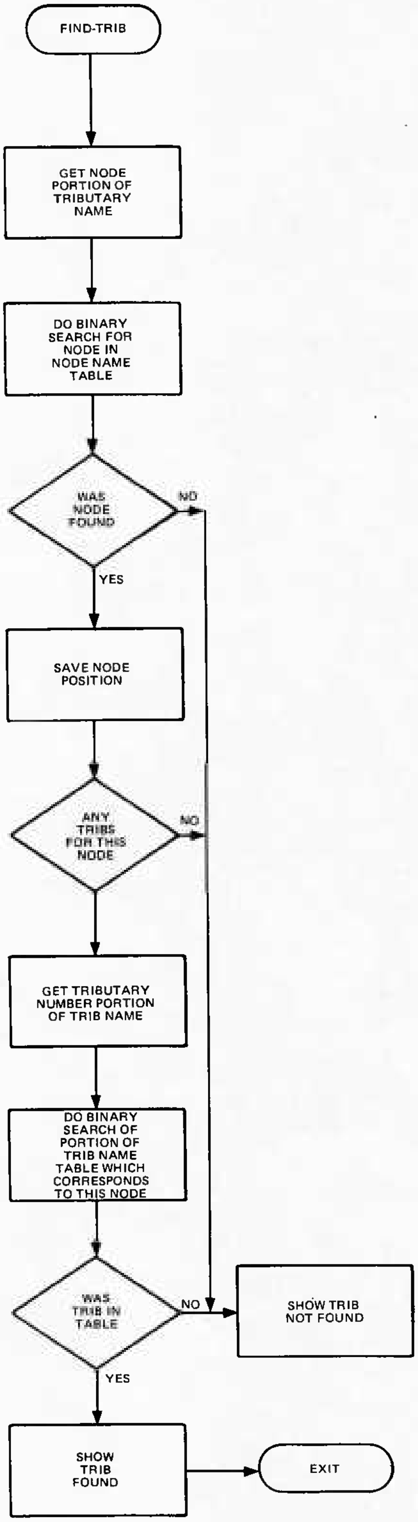


4
5A
K = K + 1 MOVE KTH DESTINATION TO OUTPUT RECORD
IS I = 9
YES
NO
4
4C
P00430
4
5C
ADD 1 TO SEQUENCE NUMBER TO SHOW CONTINUATION OF THIS MESSAGE
WRITE OUTPUT RECORD
IS LIST OPTION ON
NO
YES
TRANSMISSION TIME IN BOUNDS
NO
YES
FORMAT OUTPUT RECORD FOR PRINTING
PRINT THE RECORD
4
4B
CLEAN-UP
3, 4
5B
E.O.F ON INPUT DATA SET
YES
6
6C
NO
IS NEXT ORIGIN SAME AS THIS MESSAGE ORIGIN
YES
NO
ANY SUMMARY REPORT REQUESTED
NO
YES
IS ONLY OVERFLOW REPORT WANTED
NO
YES
ANY OVERFLOW MESSAGES THIS TRIB
NO
YES
FORMAT PARTIAL SUMMARY REPORT FOR THIS TRIB
PRINT PARTIAL REPORT
ZERO OUT SUMMARY COUNTERS
OVERFLOW SUMMARY ONLY
YES
NO
FORMAT REMAINDER OF FULL SUMMARY REPORT
PRINT REPORT
ZERO OUT REMAINDER OF SUMMARY COUNTERS
CLEANUP 20
ZERO CLOCK AND CHANNEL FREE TIMES
P00470
IS THIS 256TH DESTINATION FOR PRIOR MSG
NO
6
6A
YES
7
FIND-TRIB
SEARCH FOR DESTINATION IN NETWORK
IS TRIB IN NETWORK
YES
NO
KEEP BAD NAME FOR ERROR SUMMARY
SHOW 1 DESTINATION IN MESSAGE AND SAVE
3
3C

Discrete Message Generator Program (Sheet 5 of 7)


P00480
5
6A
GET ORIGIN OF INPUT RECORD, SHOW 0 DESTINATIONS IN MESSAGE
7
FIND-TRIB
SEARCH FOR ORIGIN IN NETWORK
IS ORIGIN IN NETWORK
NO
3
3C
YES
GET DESTINATION IN INPUT RECORD
7
FIND-TRIB
SEARCH FOR DESTINATION IN NETWORK
IS DESTINATION IN NETWORK
NO
KEEP BAD NAME FOR ERROR SUMMARY
YES
P00500
CONVERT TRANSMISSION TIME TO MILLISECONDS
SAVE ORIGIN, DESTINATION, ETC. AS CURRENT MESSAGE
SHOW 1 DESTINATION IN MESSAGE
3
3C
END-DATA
3
6B
SHOW END OF FILE ON INPUT DATA FILE
ANY DESTINATIONS IN CURRENT MESSAGE
YES
3
3D
5
6C
NO
WIND-UP
TERMINATE

Discrete Message Generator Program (Sheet 6 of 7)


FIND-TRIB
GET NODE PORTION OF TRIBUTARY NAME
DO BINARY SEARCH FOR NODE IN NODE NAME TABLE
WAS NODE FOUND
NO
YES
SAVE NODE POSITION
ANY TRIBS FOR THIS NODE
NO
GET TRIBUTARY NUMBER PORTION OF TRIB NAME
DO BINARY SEARCH OF PORTION OF TRIB NAME TABLE WHICH CORRESPONDS TO THIS NODE
WAS TRIB IN TABLE
NO
SHOW TRIB NOT FOUND
YES
SHOW TRIB FOUND
EXIT

Discrete Message Generator Program (Sheet 7 of 7)

SECTION 4 - STATISTICAL MESSAGE GENERATOR

4.1 PROGRAM REQUIREMENT DESCRIPTION

4.1.1 Program Name: Statistical Message Generator

4.1.2 Program Identification: DNSTTMG1

4.1.3 Purpose

The Statistical Message Generator program creates a message file for use by the Basic Simulation program from a set of input cards defining the statistical properties of the messages to be generated.

4.1.4 Requirements

The program was coded in FORTRAN IV. Generalized coding techniques were used to make the program as machine independent as possible. In-line subroutines using ASSIGNED GO TO statements were used where possible, rather than external subroutines to reduce program running time. The program has been initially implemented on an IBM 360/65 and requires approximately 200K bytes of memory for execution.

4.2 INPUT SPECIFICATIONS

Two input files are required by the program; the Translation and Traffic Definition files.

4.2.1 Translation File

The Translation file contains the network description. This file is created by the Status Generator program and is described in Paragraph 2.3.2.

4.2.2 Traffic Definition File

The input deck which drives the Statistical Generator consists of a series of cards which define the characteristics of the messages to be generated. The data cards are grouped into sections, each of which defines the traffic from a particular tributary to one or more other tributaries. The data defines such items as number and distribution of messages over fixed time intervals, percentage of messages for each priority and security, and probability distributions for message length and destination.

The Statistical Message Generator interprets these cards, sets up internal tables based on the data, and generates the required messages. Extensive data checking on the input data is performed. In addition, message listings and summary reports for each tributary are provided depending on control card parameters.

4.2.2.1 General Characteristics

The input is made up of a series of statements. These statements are written in free-form floating format. A parameter or data item is identified by its relative position in a string of other data items punctuated by break characters, rather than appearing in a fixed location on a card. This allows considerable freedom in punching data decks as opposed to a more restrictive format.

The break characters recognized by the Statistical Message Generator are:

Comma (,): Separates data items within a data field.

Slash (/): Separator between data fields.

Dollar Sign ($): Indicates the end of a statement.

Minus Sign (-): Always follows the statement name and separates it from the data fields which follow it.

Semicolon (;): Used only with the LENGTH statement and separates the priority identifier from the length distribution which follows.

In describing an input statement, the following rules must be observed:

All statements must be terminated by a dollar sign ($).

Blanks are always ignored.

Leading zeroes (Ø) may be omitted.

Any number of cards may be used to complete a statement.

All space on a card following a dollar sign must be blank.

Only the data of a single statement may be punched on a card.

Each statement must begin on a new card.

The following paragraphs describe the individual statements available to define the desired message characteristics.

4.2.2.2 Title for Printed Output

The first statement in the input deck may be a title card which specifies the heading to be placed at the top of every page of printed output. The full 80 columns may be used and the title should be centered on the card.

4.2.2.3 Control Statement Card

The next card in the deck (or the first if a title card is not used) is the CONTROL statement. This statement must have CONTROL in Columns 1 through 8. Other parameters in the statement may appear in any order in the statement and are defined as follows:

TIME,XXXX, YYYY

XXXX and YYYY are the beginning and end of the period over which messages are to be generated. Both parameters are specified in military time (hours and minutes). Maximum time is 36 hours (i.e., 3600).

TRAFFIC, N

This parameter provides a means to increase or decrease the number of messages generated without altering any other statement. N represents the percentage of the generated messages to be included in the simulation. Thus, when N = 100, all messages are included. For N = 50, only half are included, and for N = 200, twice the number of messages will be used. When this parameter is omitted, N is assumed to be 100.

NO MESSAGES

Bypasses generation of messages. Checks input data only.

MESSAGE NUMBER, N
Where N + 1 will be the message identification number assigned to the first message generated (optional; if omitted, N = 0 is assumed).

SEED, L
Random number seed (optional; if omitted, L = 1 is assumed).

NO OVERFLOW
Clear message queues at end of each interval (optional; if omitted, queued messages are carried over into next interval).

PRINT, I
Print the first I messages generated (optional; if omitted, individual messages are not printed).

EXACT
Generate exact number of messages per interval (optional; if omitted, interarrival times must be used. See Paragraph 6.2.7).

EDIT
No input listing or summary report is produced for a tributary unless input errors or overflow occurs.

When more than one parameter is used, they are separated by slashes.

4.2.2.4 HEADER - Switch Name/Trib Name/Minimum Intermessage Gap$

A HEADER statement must be the first statement for a tributary. All parameters are required. Both the switch and trib names are alphanumerics of at most six characters and must be consistent with the names defined in the Translation file. The minimum intermessage gap is specified in tenths of seconds and is defined as the smallest amount of time that can occur between the transmission of two successive messages on a given channel. This parameter provides a rough approximation of the operations within the tributary for removing a message from its queue and placing it on an outgoing channel.

4.2.2.5 INTERVAL-$XXX_1/XXX_2/\ldots/XXX_n$\$

The INTERVAL statements subdivide the time span of simulation into <u>n</u> intervals. Each XXX_1 is the length of the interval in minutes to a maximum of 2160 minutes (i.e., 36 hours). The sum of the intervals must equal the time span of simulation as given in the CONTROL statement. Only one INTERVAL statement per tributary is permitted. If

INTERVAL is omitted, the total time of simulation is treated as a single interval.

4.2.2.6 TYPE-TELETYPE$, TYPE-CARD$, or TYPE-MAGTAPE$

This statement specifies the message format for the messages described in the statements between the TYPE statement and its matching ENDTYPE statement. As many TYPE statements as are necessary to fully describe the traffic at the tributary may appear under one HEADER. All statements subordinate to the TYPE statement must be supplied for each additional TYPE statement. There is no limit to the number of TYPE statements for any one HEADER. All three data types may be mixed. If more than one TYPE statement is used, the INTERVAL statement must appear in the deck before the second TYPE statement and before the AVERAGE MESSAGES statement for the first TYPE statement. All statements described in Paragraphs 4.2.2.7 through 4.2.2.13 may appear in the deck following a TYPE statement in any order.

4.2.2.7 AVERAGE MESSAGES - No. of Messages/.../No. of Messages$

This statement specifies the average number of messages to be generated for each interval. There must be one entry for every interval specified on the INTERVAL statement. This applies to the EXACT option as well as the INTERARRIVAL option. This statement is required for each TYPE statement.

4.2.2.8 INTERARRIVAL - $X_1, Y_1, Y_2/.../X_n, Y_n$$

This statement defines the continuous interarrival time distribution. The X-axis number is expressed as an integer between 0 and 10000 and is interpreted as X/10000; i.e., 9000 is interpreted as 0.9 (or 90 percent) and 900 as 0.09 (or 9 percent). The Y-axis number is a time given in seconds. The X-axis values must be in ascending order and the points for X=0 and X=10000 must be defined. This statement is required only if the EXACT option is not specified in the control statement.

4.2.2.9 PRIORITY - Priority, Percent/.../Priority, Percent$

This statement defines the precedence distribution by specifying the percentage of messages for each priority or precedence level. The percentage is expressed as an integer between 0 and 10000, and is interpreted as the actual percentage times 100. That is, a percentage value of 7000 is interpreted as 70 percent. Only nonzero priorities need be specified. The total of all percentages must equal 10000 (i.e., 100 percent). This statement is required for each TYPE statement at a tributary. Precedence indicators are as follows:

Indicator	Precedence
W - Special	(1)
Z - Flash	(2)
O - Immediate	(3)
P - Priority	(4)
R - Routine	(5)

4.2.2.10 LENGTH - Priority; X_1, Length/X_2, Length/.../X_n, Length$

The LENGTH statement defines the continuous message-length distribution according to priority level. The same number of LENGTH cards must be used as there are priority levels specified on the PRIORITY card. The X-axis number is expressed as an integer between 0 and 10000 and is interpreted as X/10000. The length may be any integer less than 5000 and is interpreted as tenths of line blocks. The X-values must be in ascending order and the points for X=0 and X=10000 must be defined.

The priority indicators accepted by the program are the same as those listed under the PRIORITY statement.

4.2.2.11 SECURITY - Security, Percent/.../Security, Percent$

The SECURITY statement defines the security distribution by specifying the percentage of messages that are of a certain security. The percentages are interpreted in the same manner as the PRIORITY statement. Total percentage must equal 10000 (100 percent). If this card is omitted, all messages will be of the lowest security level. The security indicators recognized by the program are:

A - Special

T - Top Secret

S - Secret

C - Confidential

U - Unclassified

4.2.2.12 NUMDEST - No. of Destinations, Percent/.../No. of Destinations, Percent$

The NUMDEST statement defines the number of destinations distribution. Each set of parameters defines the percentage of messages that will have the indicated number of destinations. The total must equal 100 percent. If this card is omitted, all messages will have a single destination.

4.2.2.13 WEIGHT DESTINATIONS - Destination, Percent/.../Destination, Percent$

The WEIGHT DESTINATIONS statement determines the distribution of destinations. Each set of parameters defines the percentage of messages to be delivered to a specific destination. The destination is a one to six character alphanumeric tributary name, which must be consistent with those defined in the Translation file. The total percentage must equal 100 percent. If this card is omitted, all destinations have equal weight.

4.2.2.14 ENDTYPE$

The ENDTYPE cards indicate that all information for a given type at a particular tributary has been given. This card should only be used if information for an additional type at the same tributary follows.

4.2.2.15 ENDTRIB$

The ENDTRIB card indicates that all information for a given tributary has been specified.

4.3 OUTPUT SPECIFICATION

Program output consists of printed information and a traffic file containing the messages generated by the program.

4.3.1 Printed Output

Depending on options specified on the CONTROL statement given in the Traffic Definition file, various reports are printed. Output is organized on a tributary-by-tributary basis.

If the EDIT parameter was not specified on the CONTROL statement in the Traffic Definition file, all statements in that file pertaining to the tributary are printed. In addition, after the messages for the file are generated, a report summarizing the characteristics of these messages is printed. If the EDIT option is specified, these two reports are printed for the tributary only if there were input errors or if overflow occurred during the message generation process. Normally, a one-line report indicating the number of messages generated for the tributary is printed.

Regardless of the EDIT parameter, if the PRINT, I parameter is present, the first I messages generated are reformatted in a form suitable for visual interpretation and printed. This output is interleaved with the reports.

4.3.2 Traffic File

The messages generated by this program are identical to these produced by the Discrete Message Generator. The format of these messages is the same as those described in Paragraph 3.3.

4.4 PROGRAM LOGIC

4.4.1 Program Initialization

The first step in program execution reads in the CONTROL statement and evaluates its parameter fields. This statement is free-form, and the parameters can be mixed in any order. The parameter keywords, and any associated data items, are examined one at a time, checked against a list of allowable parameters, and appropriate variables are set in the program. The entire statement is checked regardless of errors encountered in any particular field. If any errors are encountered, the program terminates with a completion code of 8 and no messages are generated.

Next, pertinent information from the Translation file is read into core. The switch names are extracted from the Component cards, translated into integer numbers, and placed into array NODES (I). The character versions of these names are packed in A4, A2 format into array NNAMES (2, I). Any link cards in the file are skipped. Tributary names, number of channels, and transmission speeds are taken from the Tributary cards. The names are translated into unique integer values and put into array TRNAME (I). The character versions of the names go into array TNAMES (2, I). The channels are placed into array NCHANS (I) and the teletype, card, and mag-tape transmission delays are put into array SPEEDS (3, I), all in the same relative position as the tributary names.

The HEADER and WEIGHT DESTINATIONS statements contain tributary names which must be verified as belonging in the network described in the Translation file. By converting these names to unique integer values, table searching can be made more effective as it is necessary to compare only one integer rather than a string of characters. The technique for translating names to numbers is described in Paragraph 2.4.8.

Since there may be a large number of tributaries in the network, table searching can be quite expensive in terms of running time. Accordingly, the tributary names and their associated data are put into sorted order via a simple bubble sort. For normal runs the tributaries are already in sorted order in the Translation file, but if a network with nonstandard naming conventions is being simulated this may not be the case, hence the precautionary sort.

If any error is found in the Translation file the program terminates with a completion code of 8.

4.4.2 Input Processing

The Statistical Message Generator operates in two phases. First, the input statements for one tributary are read into core and evaluated. The statistical data is converted into internal tables and constants. When all necessary data has been assembled, the output messages are generated. This paragraph describes the input processing phase.

The input is free-format in terms of statement ordering and length. There are some restrictions on statement ordering, but for the most part they may appear in any order. There are few restrictions on statement lengths, making it almost impossible to preallocate storage for a particular kind of statement. This is further complicated since there is not a fixed number of statements to be handled.

Accordingly, storage space for the input data must be allocated on a dynamic basis. Array WORK (I) is used to contain the statistical tables and constants found in the input. A system of pointers, arranged in levels of subordination, is used to keep track of the information. At the first level is the HEADER. There is only one HEADER for the tributary, and the first 10 words of array WORK (I) are assigned to it. The HEADER points to the INTERVAL work area and to the work area belonging to the first TYPE statement for each of the three data types. These work areas may start at any point in WORK, depending on their position in the input file.

The TYPE statements constitute the second level of pointers. Each TYPE area points to the next TYPE area of its own data type (if any) and to the work areas of all other statements appearing in the input deck between the TYPE statement and its associated ENDTYPE statement.

As each statement is processed, its data items are placed in contiguous locations in array WORK, starting at the next available location in the array. The statement's address is defined as the location of its first data item relative to the beginning of the array. The work area address is inserted in the appropriate location in the work area of the statement to which it is subordinate.

Briefly, the program works in the following manner. Processing for a tributary begins with its HEADER statement. Any cards between an ENDTRIB statement and the next HEADER are ignored. The statement following the HEADER must be either an INTERVAL or TYPE statement. Any other is treated as an error. If the next card is an INTERVAL statement, the statement following must be a TYPE statement.

The statements following the TYPE statement may be any of those described in Paragraphs 4.2.2.7 through 4.2.2.13. If the INTERVAL statement did not precede the TYPE statement, it can also be mixed in with the others. However, it must appear before the AVERAGE MESSAGES statement for the first TYPE statement.

Each statement ends with a dollar sign, and the next statement begins on the next card in the file. Once the statement is identified, one of a number of routines is invoked to process it.

When an ENDTYPE statement is encountered, the program looks for the next TYPE statement. This procedure continues until an ENDTRIB statement occurs in the input deck. At this point, if there were no errors in the data for this tributary, subroutine MESGEN is called to generate traffic defined by the input statements. The program now expects the next HEADER. The entire cycle repeats until an end of file is encountered in the Traffic Definition file.

4.4.2.1 Data Fields

The basic driver for the program is the data field. A data field is defined as any group of characters between a pair of break characters or between the start of a statement and the first break character.

An in-line subroutine (identified as SUB-70 in the flow charts) is used to extract the next data field from a statement. Variable COLUMN points to the last card column processed on the current input card. Array FIELD (80) contains the data field. Variable NCHAR contains the length of the data field and variable BREAK contains the break character which terminates the field.

Starting at location COLUMN +1 on the input card (contained in A1 format in array NCARD (80))and location 1 in array FIELD, characters are extracted from the input card and put into FIELD. Blank characters are ignored. This process continues until a break character is encountered. Variable BREAK is then set to a number value according to:

/ = 1

$ = 2

, = 3

; = 4

- = 5

Variable ENDCRD is used to indicate various end conditions. It is assigned values according to the scheme:

0 = No end condition

1 = End of statement

2 = ENDTYPE card

3 = ENDTRIB card

4 = End of file

If the break character for this field is a $, ENDCRD is set to 1.

If, in the process of extracting the next data field, the end of the current input card is reached without finding a break character, a new card is read into array NCARD and processing continues as before. As each new card is read, a precautionary check is made to ensure that the new card is not an ENDTYPE or ENDTRIB statement. This also ensures that syntax errors in one TYPE or HEADER sequence will not affect the following one. If this was an ENDTYPE or ENDTRIB, or if an end of file was encountered, variable ENDCRD is set to the appropriate value. In addition, variable BREAK is set to 2 effectively generating the missing dollar sign for the previous statement.

4.4.2.2 Statement Identification

The Statistical Message Generator knows when it is expecting a new statement. It is either looking for a new HEADER or TYPE statement following an ENDTYPE or ENDTRIB card, or a dollar sign was found on an input card, indicating the termination of that statement. A new statement should begin on the next card in the file.

Variable COLUMN is set to 80 and a branch is made to SUB-70, returning the first field on the next card. This should be the statement name, with a minus sign break character. The statement is identified by checking all legal names of the same length as the data field. Once identified, the remainder of the statement is processed by one of several in-line routines which evaluates its particular data items. Due to the similarity of statements, fewer routines are required than there are kinds of statements. These routines are not treated in detail here, but their general mode of operation is described in the next paragraph.

4.4.2.3 Statement Processing

The first step taken when a statement is identified is to establish a work area for it. SUB-76 (as identified in the flow charts) is used for this purpose. The next available location in array WORK is assigned to the statement, and a pointer to the area is established in the work area of the statement to which this one is subordinate. Variable CHAIN always points to the next free word in the array.

The first word in the work area is reserved for use as a pointer or counter. Variable NEXTWD is set to the location of the second word of the work area. NEXTWD will be used as the base for inserting any data. If this is a TYPE statement an entire 31-word work area is obtained at once. SECURITY and PRIORITY statements do not use SUB-76, since their information is kept within the TYPE work area itself.

Aside from the TYPE, SECURITY, and PRIORITY statements, all of the statements follow one of two basic formats.

1. $-X_1/X_2/\ldots/X_n\$$
2. $-X_1, Y_1/X_2, Y_2/\ldots/X_n, Y_n\$$

In the first case one item at a time is evaluated and inserted into the work area. In the second case each X, Y pair is evaluated prior to insertion into the work area. Variable NWORDS is set to either 1 or 2 depending on how many words are being added to the work area. Array VALUES (2) is used to temporarily hold the value(s) being inserted. Sub-76 is called to put the items into array WORK. In this manner absolute control can be maintained over available storage.

Extensive error checking is performed on each statement. Syntax must be exact. However, if at all possible, the program will recover from syntax errors and process as much of each statement as possible. Individual data items are checked to assure that their values conform to the requirements in Paragraph 4.2.1.

After processing a statement, any percentages in the data for the statement are converted into their cumulative form.

4.4.2.4 Work Area Layout

The HEADER area for a tributary always occupies the first 10 words of array WORK. The content of each of these words is given in Table 4-1.

Each TYPE statement is assigned a 31-word work area which is pointed to from either the HEADER area or the previous TYPE area for this particular data type. The format of this work area is described in Table 4-2.

The work area for the INTERVAL statement is laid out in Table 4-3. The extra spaces are used to keep summary statistics for each interval during message generation.

The AVERAGE MESSAGES work area is similar to that for an INTERVAL statement; however it is not padded. Its format is given in Table 4-4.

INTERARRIVAL, WEIGHT DESTINATIONS, LENGTH, and NUMDEST statements all use work areas of similar format, although their content may differ. Their general format is:

Statement Name - X_1, Y_1/X_2, Y_2/.../X_n, Y_n\$

Their core representation is shown in Table 4-5.

When array WORK has been filled for a particular tributary, it is passed to subroutine MESGEN and used to drive the message generation process.

4.4.3 Message Generation - Subroutine MESGEN

Message generation is performed on an interval basis. Within each interval two passes are required to create the messages. On the first pass the transmission times for the messages are established in message prototypes. The second pass sorts these into a time-ordered list and the remainder of the fields are filled for each message.

4.4.3.1 Message Generation - Pass 1

As processing begins it is necessary to locate the INTERVAL work area. Its address is stored in WORK (5). Variable BASE1 is established with this address. To find the length of the I-th Interval, the program references WORK (BASE1 + I).

The program operates one interval at a time until all intervals have been handled. As an interval is processed, each TYPE area is taken in turn. BASE2 is set up with the starting address of the particular TYPE area. BASE3 is established with the location of the AVERAGE MESSAGES area for the TYPE area in question. It is found in WORK (BASE2 + 1). To find the number of messages to be generated for this TYPE area in the I-th Interval, WORK (BASE3 + 1) is referenced. Knowing the length of the interval and the number of messages to be generated for the TYPE area, message prototypes can be created for this individual TYPE area.

Each prototype consists of two words. The first word contains the transmission time for the message and the second is filled with the current value of BASE2. This allows the program to reference the proper work area during Pass 2.

Times are determined in one of two ways, depending on the CONTROL statement. If the EXACT option was specified (the normal mode of operation), each time is established by merely drawing a random integer between zero and the interval length. If the exact option is not on, the times must be taken from the interarrival distribution. The INTERARRIVAL work area is set up as a cumulative probability distribution, with the X-axis values running from 0 to 10000 (representing 0 to 100 percent). Each time is determined by drawing a random integer between 0 and 10000 and using simple linear interpolation to arrive at the time. Each time is interpreted to mean the time from the beginning of the interval rather than from the time of the last message.

Each prototype is placed in the next available location in array MSGBUF (2,I). When all messages for all TYPE areas in this interval have been put into MSGBUF, control goes to Pass 2.

4.4.3.2 Message Generation - Pass 2

The first step at this stage of processing is to sort the message prototypes into a time-ordered sequence. The sorting technique is simple. The minimum time is located, and the time and its associated TYPE area address are put in the first location in MSGBUF (2, I). Then, the next smallest time is found and put in the second location, and so on, until the message buffer has been arranged in ascending time order.

The actual messages can now be generated for this interval. The program takes each message prototype in turn, establishes BASE2 with the appropriate TYPE area address and randomly generates the message's characteristics.

Before generating any message parameters, the message time-of-transmission must be established. The program keeps an internal clock and also keeps track of the availability times for each of the channels at the current tributary. These channel times are stored in array CHFREE (I). As each message is examined, the clock is set to the transmission time in the prototype or the time when the next channel becomes available, whichever is greater.

If the transmission time is greater than the end of the interval, this message and any following it in MSGBUF are considered to be overflow messages. All of these overflow messages are copied into OVFBUF (2, I) and their times reset to zero. When the next interval is processed, any messages in OVFBUF will be generated before those in MSGBUF. Once the messages have been copied into OVFBUF, control goes to Pass 1 and the next interval.

Presuming that the transmission time was in bounds, the message is assigned the next sequential message number. Its priority and security are randomly generated in such a manner as to ensure that the percentage of messages generated with each priority or security level are exactly as specified on the PRIORITY or SECURITY statements (subject to round-off error).

This technique is as follows. Assume that a PRIORITY statement has been used which specifies equal probability of 20 percent for each priority. Also, assume that for a particular interval 10 messages are to be generated for the TYPE area containing this priority information. During Pass 1, locations 22 thru 26 of the TYPE area are initialized with the cumulative number of messages to be generated for each priority; e.g.

Priority	W	Z	O	P	R
Cumulative Messages	2	4	6	8	10

During Pass 1, 10 message prototypes were created with the TYPE area address. As each one is processed, the above table is used to determine the priority.

For the first such message a random number is drawn between 1 and 10. A scan is then made from left to right to find the interval containing the number. A random number of 7 or 8 would result in a priority of P. If the random number 4 was drawn the priority becomes Z. The interval count is decremented by 1, as are all intervals to the right of it, thus maintaining a cumulative table. At this point the table becomes:

Priority	W	Z	O	P	R
Cumulative Messages	2	3	5	7	9

For the next message (for this TYPE area) a random number is drawn between 1 and 10, as before. If a 3 were drawn, the priority would again be Z and the table would become:

Priority	W	Z	O	P	R
Cumulative Messages	2	2	4	6	8

At this point it is impossible to assign a priority Z to another message, since a random number of 2 will cause assignment of a priority W. This process continues until the table becomes all zeroes. Next, the message length is assigned. The LENGTH table associated with the priority generated is used. The table is set up as a continuous probability distribution, with the X-axis running from 0 to 10000 (representing 0 to 100 percent). A random number is drawn between 0 and 10000 and simple linear interpolation is used to obtain the length. When the length is known, the message transmission time is calculated and the channel assigned to the message is made unavailable for that period of time.

The number of destinations is a discrete rather than continuous distribution, again with the X-axis representing a 100 percent span. A random number between 0 and 10000 is drawn, a simple scan is made to locate the interval into which it falls, and the number of destinations is directly read out. A loop is set up using the number of destinations as a counter. The WEIGHT DESTINATIONS work area also is a discrete distribution. Each destination is generated and added to the message.

Depending on the number of destinations, one or more message records are written onto file MESOUT. The first record for the message will have up to five destinations. Successive records will have up to nine destinations. If the LIST option is on, the message will also be reformatted and printed.

Message generation continues until all messages in MSGBUF have been processed or an overflow occurs. Control then goes to Pass 1 for handling of the next interval.

If this is the last interval, a summary report may be written. If the EDIT option is on, the summary report is written only if there was an overflow in some interval during message generation; otherwise the report is automatically written. The report shows statistics for the messages as actually generated and it may be determined whether they have the desired characteristics.

Flow charts of the program logic are included at the end of this section.

4.5 MACHINE DEPENDENCE

This program was implemented as a generalized subset of the FORTRAN IV programming language. It should be machine independent except for minor compiler differences. Hardware requirements are that the machine on which it is run should have a word length of 32 bits or more, and that a word will contain four characters or more of alphanumeric data.

Table 4-1. HEADER Statement Work Area

Word Number	Content
1	Pointer to Teletype TYPE Area*
2	Pointer to Card TYPE Area*
3	Pointer to Mag-Tape TYPE Area*
4	Number of Intervals
5	Pointer to INTERVAL Area
6	Position of Switch Name in Array NNAMES (2, I)
7	Position of Tributary Name in Array TNAMES (2, I)
8	Intermessage Gap
9-10	Ø - Unused

*Set to Zero if Unused

Table 4-2. TYPE Statement Work Area

Word Number	Content
1	Pointer to next TYPE Area of this type*
2	Pointer to AVERAGE MESSAGES Area*
3	Pointer to INTERARRIVAL Area*
4-8	PRIORITY Percentages (W, Z, O, P, R)
9-13	Pointers to Length Area for Each Priority*
14-18	SECURITY Percentages (A, T, S, C, U)
19	Pointer to NUMDEST Area*
20	Pointer to WEIGHT DESTINATIONS Area*
21	Data Type 1 = Teletype 2 = Data 3 = Mag Tape
22-26	Work space for generation of Priorities. Initialized at all zeroes.
27-31	Work space for generation of Security. Initialized at all zeroes.

*Set to zero if unused

Table 4-3. INTERVAL Statement Work Area

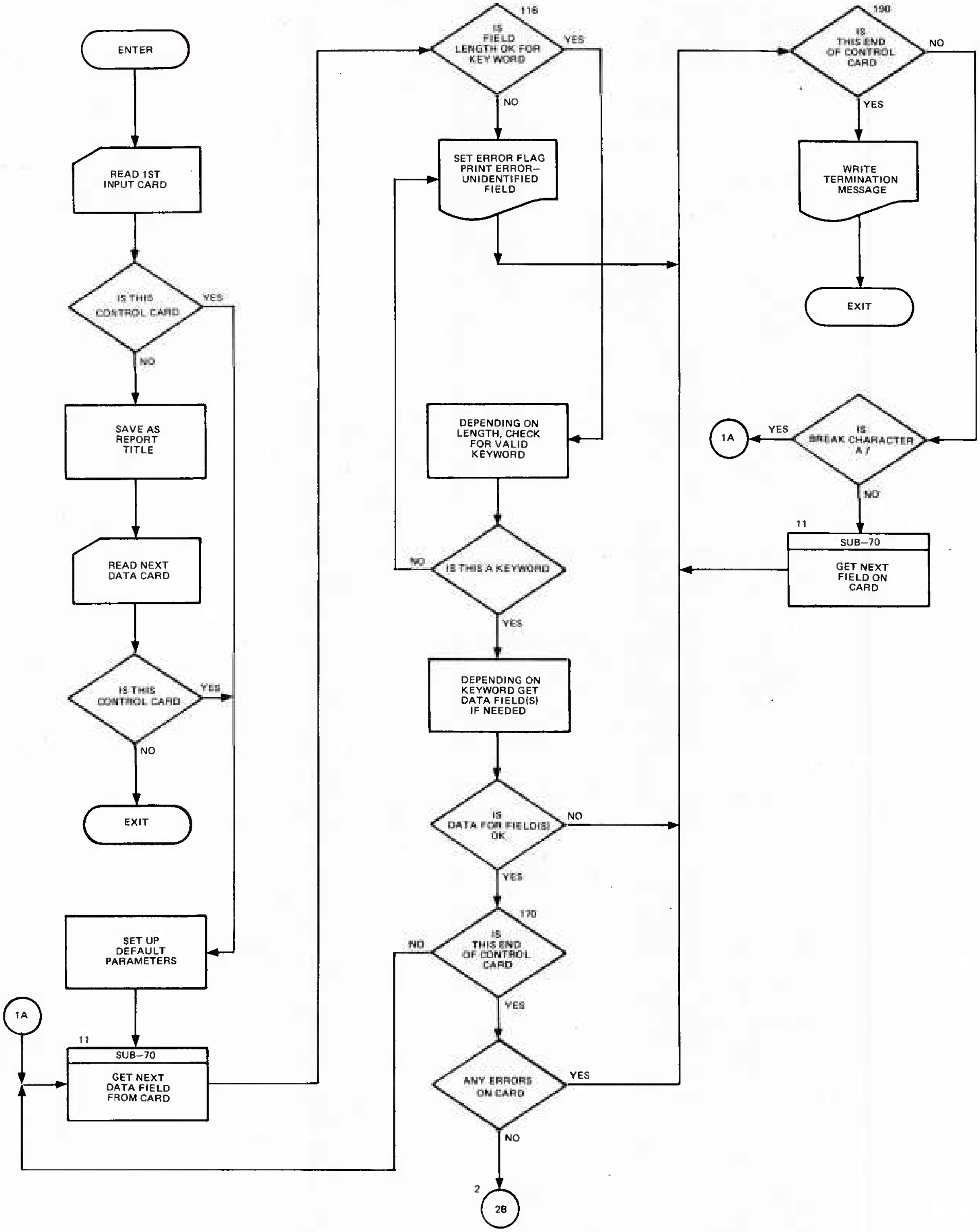
Word Number	Content
1	0 - Unused
2	Length of 1st Interval
3	0
4	0
5	Length of 2nd Interval
6	0
7	0
.	.
.	.
.	.
3*N-1	Length of Nth Interval
3*N	0
3*N+1	0

Table 4-4. Average Messages Work Area

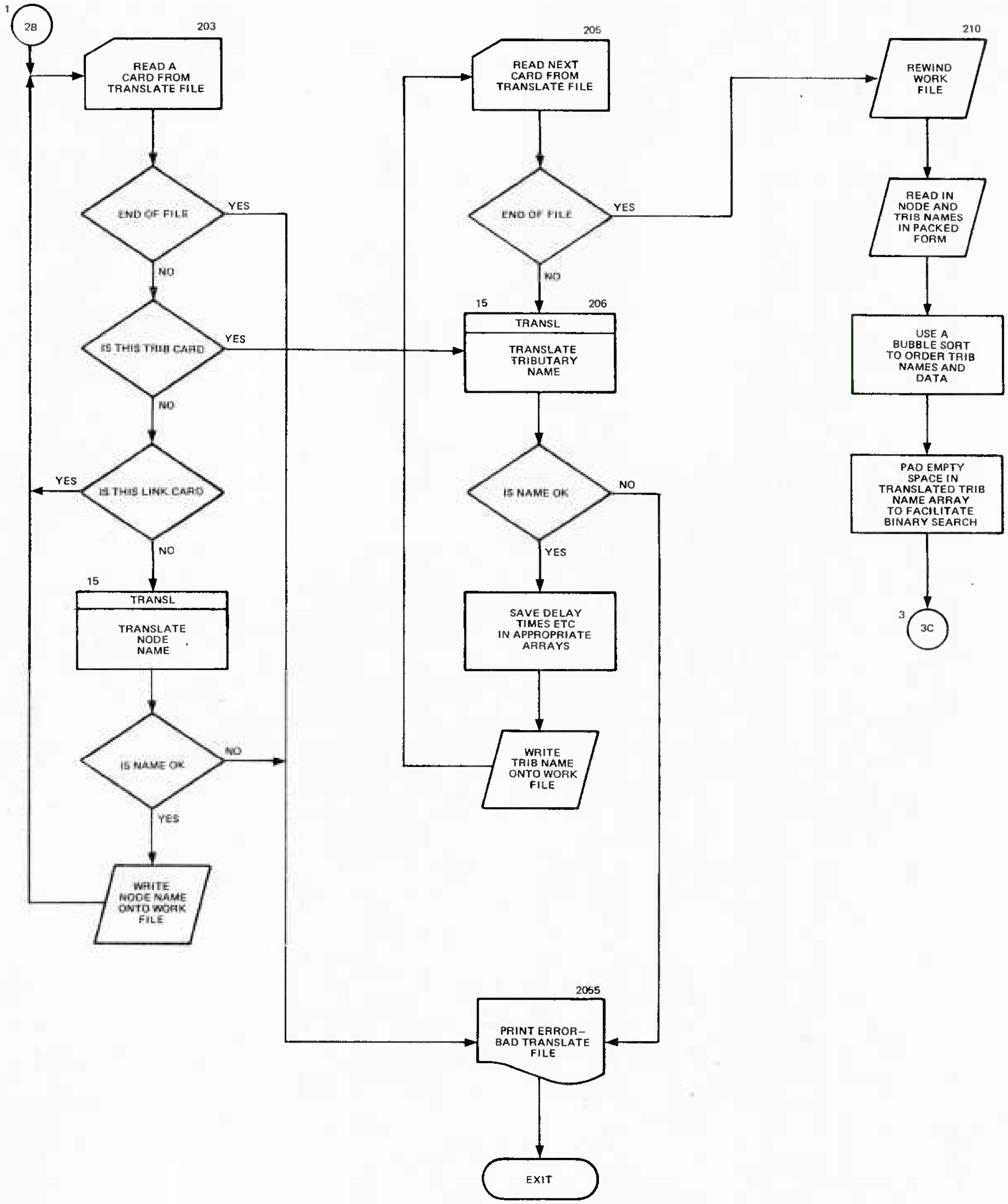
Word Number	Content
1	0 - Unused
2	Number of messages for 1st Interval
3	Number of messages for 2nd Interval
.	.
.	.
.	.
N+1	Number of Messages for Nth Interval

Table 4-5. Work Area Format for INTERVAL, WEIGHT DESTINATIONS, LENGTH, or NUMDEST Statement

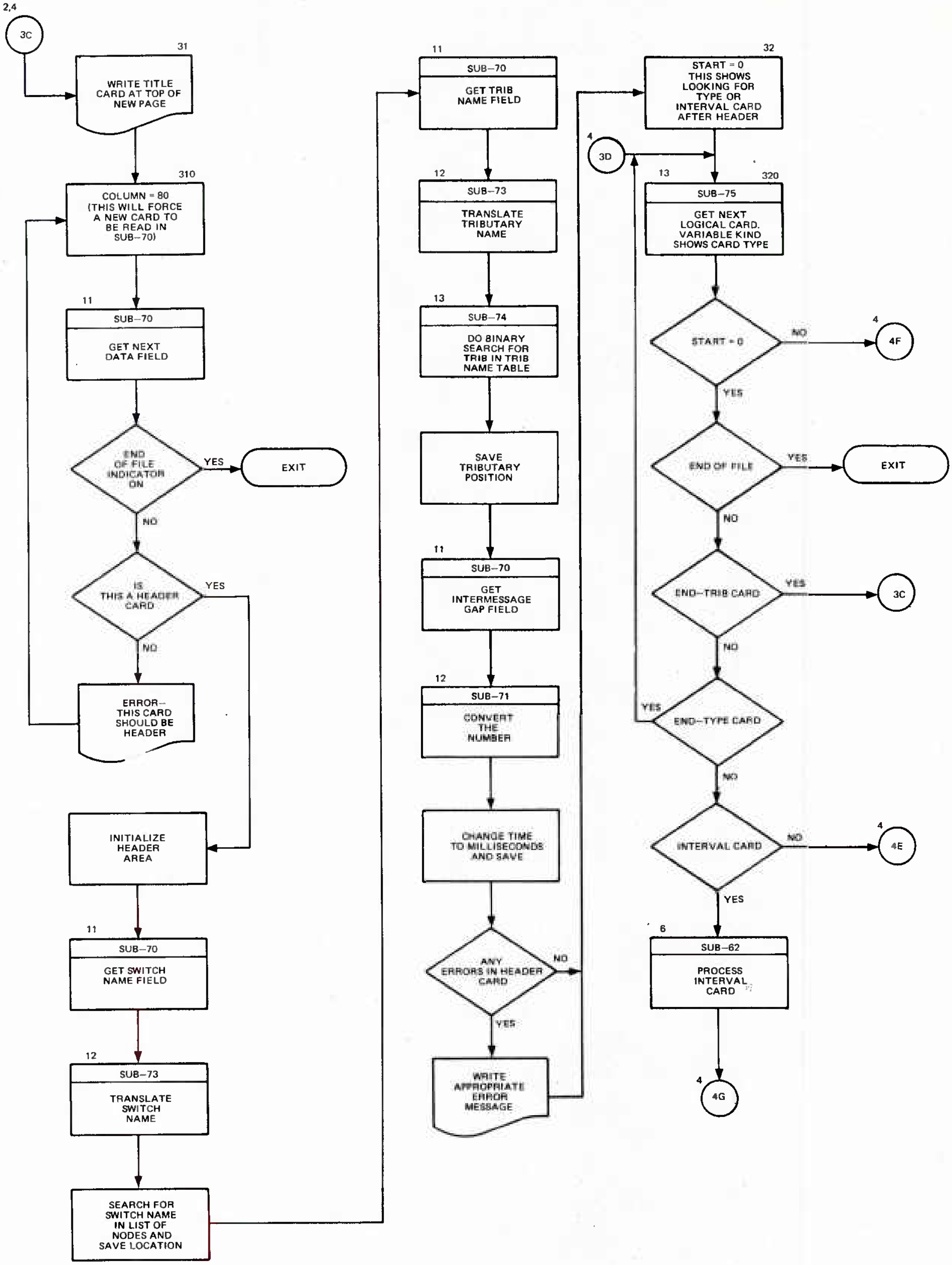
Word Number	Content
1	Number of Data Pairs following (i. e. N0)
2	1st X - Value
3	1st Y - Value
4	2nd X - Value
5	2nd Y - Value
.	.
.	.
.	.
2*N	Nth X - Value
2*N+1	Nth Y - Value


ENTER
READ 1ST INPUT CARD
IS THIS CONTROL CARD
YES
NO
SAVE AS REPORT TITLE
READ NEXT DATA CARD
IS THIS CONTROL CARD
YES
NO
EXIT
SET UP DEFAULT PARAMETERS
1A
11
SUB–70
GET NEXT DATA FIELD FROM CARD
116
IS FIELD LENGTH OK FOR KEY WORD
YES
NO
SET ERROR FLAG PRINT ERROR– UNIDENTIFIED FIELD
DEPENDING ON LENGTH, CHECK FOR VALID KEYWORD
NO
IS THIS A KEYWORD
YES
DEPENDING ON KEYWORD GET DATA FIELD(S) IF NEEDED
IS DATA FOR FIELD(S) OK
NO
YES
170
NO
IS THIS END OF CONTROL CARD
YES
ANY ERRORS ON CARD
YES
NO
2
2B
190
IS THIS END OF CONTROL CARD
NO
YES
WRITE TERMINATION MESSAGE
EXIT
1A
YES
IS BREAK CHARACTER A /
NO
11
SUB–70
GET NEXT FIELD ON CARD

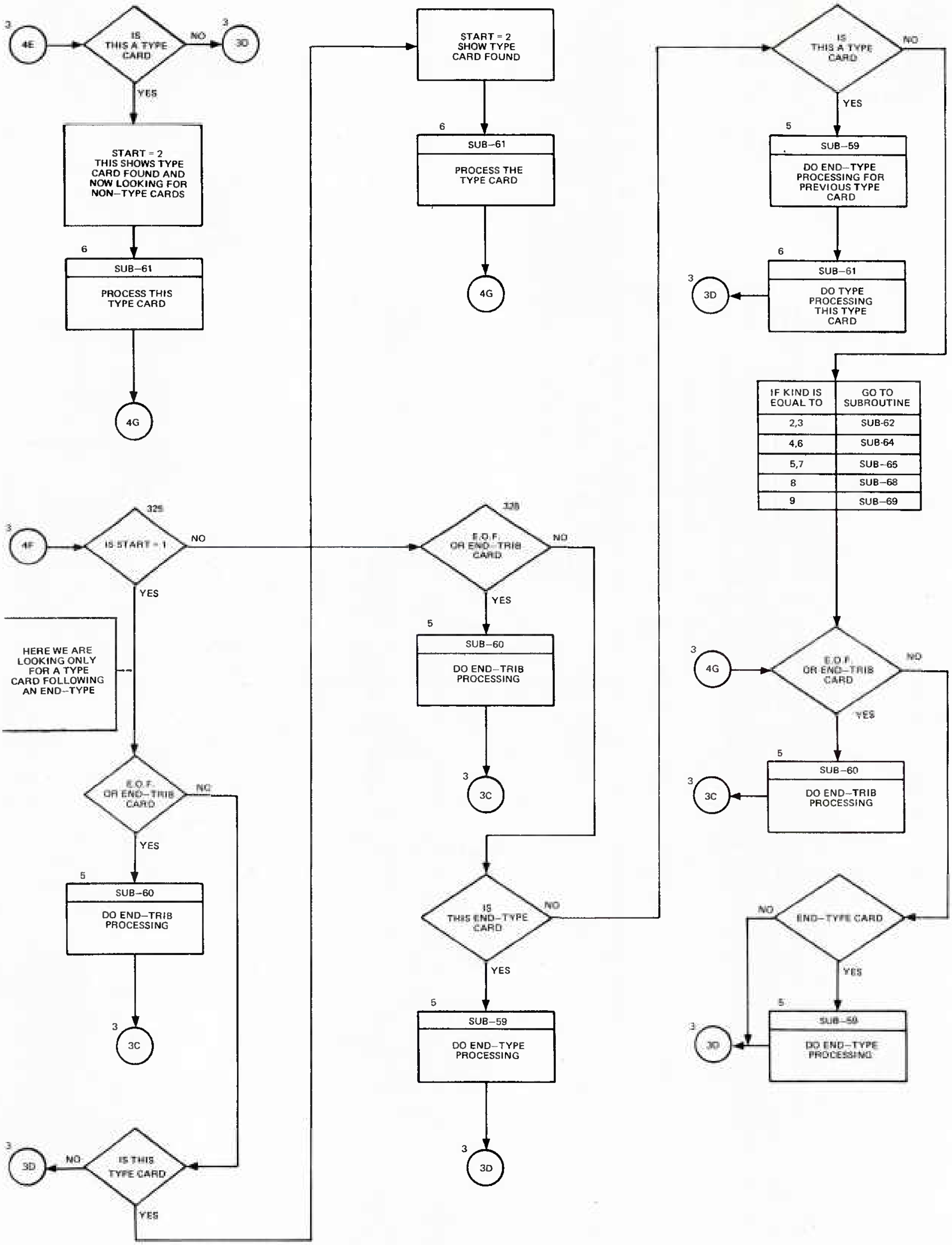
Statistical Message Generator Program (Sheet 1 of 17)


1
2B
203
READ A CARD FROM TRANSLATE FILE
END OF FILE
YES
NO
IS THIS TRIB CARD
YES
NO
IS THIS LINK CARD
YES
NO
15
TRANSL
TRANSLATE NODE NAME
IS NAME OK
NO
YES
WRITE NODE NAME ONTO WORK FILE
205
READ NEXT CARD FROM TRANSLATE FILE
END OF FILE
YES
NO
15
206
TRANSL
TRANSLATE TRIBUTARY NAME
IS NAME OK
NO
YES
SAVE DELAY TIMES ETC IN APPROPRIATE ARRAYS
WRITE TRIB NAME ONTO WORK FILE
2055
PRINT ERROR– BAD TRANSLATE FILE
EXIT
210
REWIND WORK FILE
READ IN NODE AND TRIB NAMES IN PACKED FORM
USE A BUBBLE SORT TO ORDER TRIB NAMES AND DATA
PAD EMPTY SPACE IN TRANSLATED TRIB NAME ARRAY TO FACILITATE BINARY SEARCH
3
3C

Statistical Message Generator Program (Sheet 2 of 17)

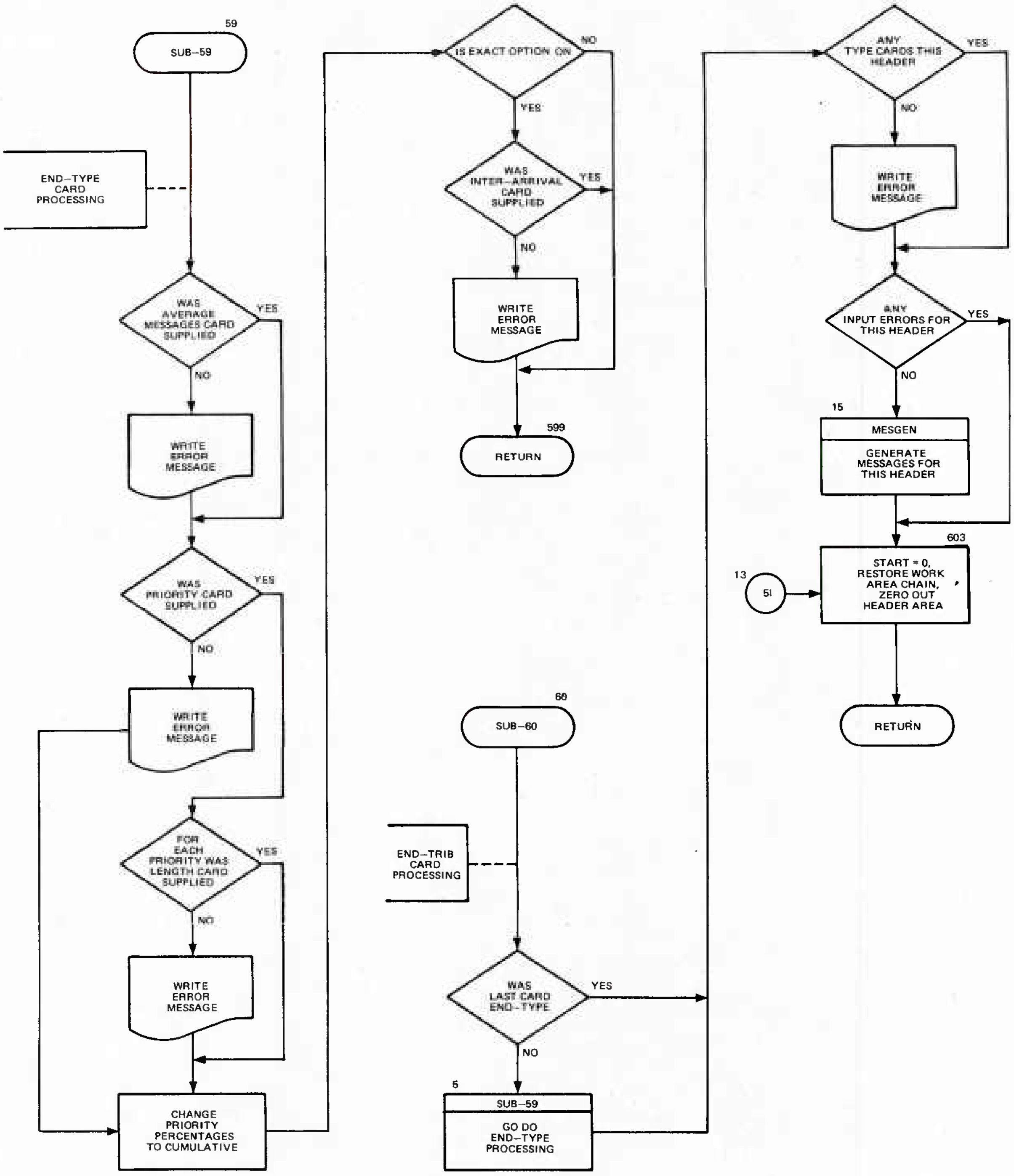

2,4
3C
31
WRITE TITLE CARD AT TOP OF NEW PAGE
310
COLUMN = 80 (THIS WILL FORCE A NEW CARD TO BE READ IN SUB–70)
11
SUB–70
GET NEXT DATA FIELD
END OF FILE INDICATOR ON
YES
EXIT
NO
IS THIS A HEADER CARD
YES
NO
ERROR– THIS CARD SHOULD BE HEADER
INITIALIZE HEADER AREA
11
SUB–70
GET SWITCH NAME FIELD
12
SUB–73
TRANSLATE SWITCH NAME
SEARCH FOR SWITCH NAME IN LIST OF NODES AND SAVE LOCATION
11
SUB–70
GET TRIB NAME FIELD
12
SUB–73
TRANSLATE TRIBUTARY NAME
13
SUB–74
DO BINARY SEARCH FOR TRIB IN TRIB NAME TABLE
SAVE TRIBUTARY POSITION
11
SUB–70
GET INTERMESSAGE GAP FIELD
12
SUB–71
CONVERT THE NUMBER
CHANGE TIME TO MILLISECONDS AND SAVE
ANY ERRORS IN HEADER CARD
NO
YES
WRITE APPROPRIATE ERROR MESSAGE
32
START = 0 THIS SHOWS LOOKING FOR TYPE OR INTERVAL CARD AFTER HEADER
4
3D
13
320
SUB–75
GET NEXT LOGICAL CARD. VARIABLE KIND SHOWS CARD TYPE
START = 0
NO
4
4F
YES
END OF FILE
YES
EXIT
NO
END–TRIB CARD
YES
3C
NO
END–TYPE CARD
YES
NO
INTERVAL CARD
NO
4
4E
YES
6
SUB–62
PROCESS INTERVAL CARD
4
4G

Statistical Message Generator Program (Sheet 3 of 17)

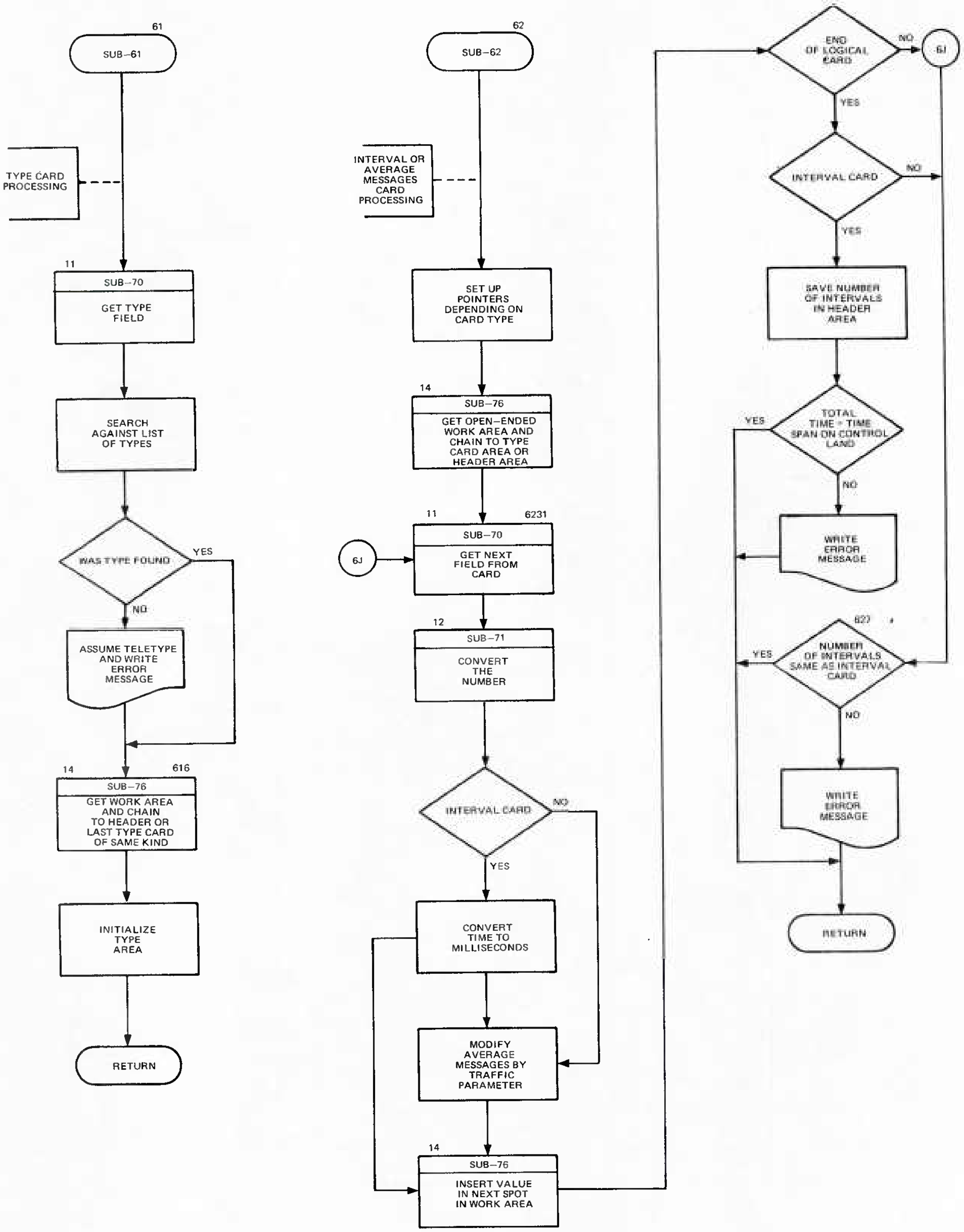


IF KIND IS EQUAL TO	GO TO SUBROUTINE
2,3	SUB-62
4,6	SUB-64
5,7	SUB-65
8	SUB-68
9	SUB-69

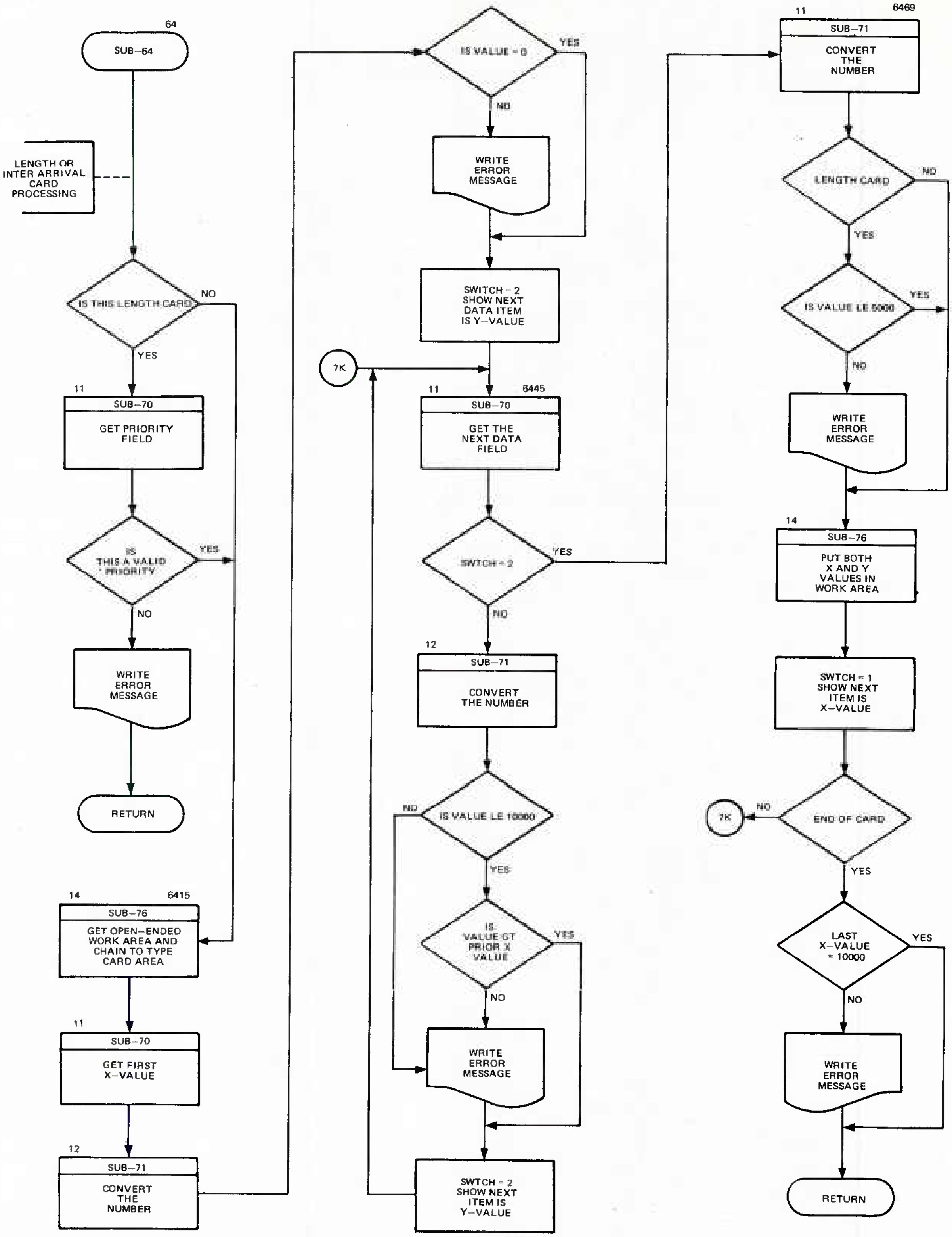
Statistical Message Generator Program (Sheet 4 of 17)


59
SUB–59
END–TYPE CARD PROCESSING
WAS AVERAGE MESSAGES CARD SUPPLIED
YES
NO
WRITE ERROR MESSAGE
WAS PRIORITY CARD SUPPLIED
YES
NO
WRITE ERROR MESSAGE
FOR EACH PRIORITY WAS LENGTH CARD SUPPLIED
YES
NO
WRITE ERROR MESSAGE
CHANGE PRIORITY PERCENTAGES TO CUMULATIVE
IS EXACT OPTION ON
NO
YES
WAS INTER–ARRIVAL CARD SUPPLIED
YES
NO
WRITE ERROR MESSAGE
599
RETURN
60
SUB–60
END–TRIB CARD PROCESSING
WAS LAST CARD END–TYPE
YES
NO
5
SUB–59
GO DO END–TYPE PROCESSING
ANY TYPE CARDS THIS HEADER
YES
NO
WRITE ERROR MESSAGE
ANY INPUT ERRORS FOR THIS HEADER
YES
NO
15
MESGEN
GENERATE MESSAGES FOR THIS HEADER
603
13
51
START = 0, RESTORE WORK AREA CHAIN, ZERO OUT HEADER AREA
RETURN

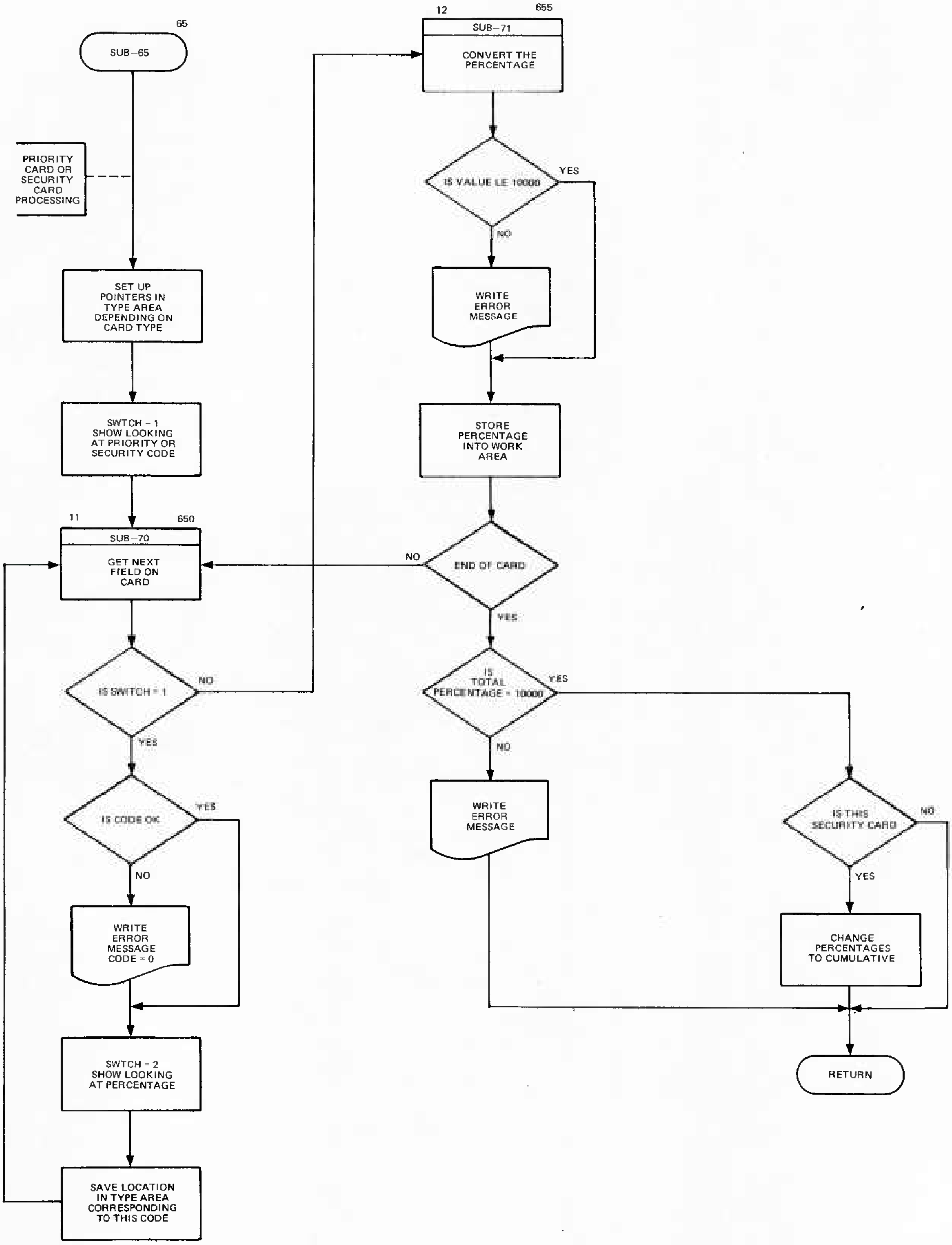
Statistical Message Generator Program (Sheet 5 of 17)


61
SUB–61
TYPE CARD PROCESSING
11
SUB–70
GET TYPE FIELD
SEARCH AGAINST LIST OF TYPES
WAS TYPE FOUND
YES
NO
ASSUME TELETYPE AND WRITE ERROR MESSAGE
14
616
SUB–76
GET WORK AREA AND CHAIN TO HEADER OR LAST TYPE CARD OF SAME KIND
INITIALIZE TYPE AREA
RETURN
62
SUB–62
INTERVAL OR AVERAGE MESSAGES CARD PROCESSING
SET UP POINTERS DEPENDING ON CARD TYPE
14
SUB–76
GET OPEN–ENDED WORK AREA AND CHAIN TO TYPE CARD AREA OR HEADER AREA
11
6231
6J
SUB–70
GET NEXT FIELD FROM CARD
12
SUB–71
CONVERT THE NUMBER
INTERVAL CARD
NO
YES
CONVERT TIME TO MILLISECONDS
MODIFY AVERAGE MESSAGES BY TRAFFIC PARAMETER
14
SUB–76
INSERT VALUE IN NEXT SPOT IN WORK AREA
END OF LOGICAL CARD
NO
6J
YES
INTERVAL CARD
NO
YES
SAVE NUMBER OF INTERVALS IN HEADER AREA
TOTAL TIME = TIME SPAN ON CONTROL LAND
YES
NO
WRITE ERROR MESSAGE
627
NUMBER OF INTERVALS SAME AS INTERVAL CARD
YES
NO
WRITE ERROR MESSAGE
RETURN

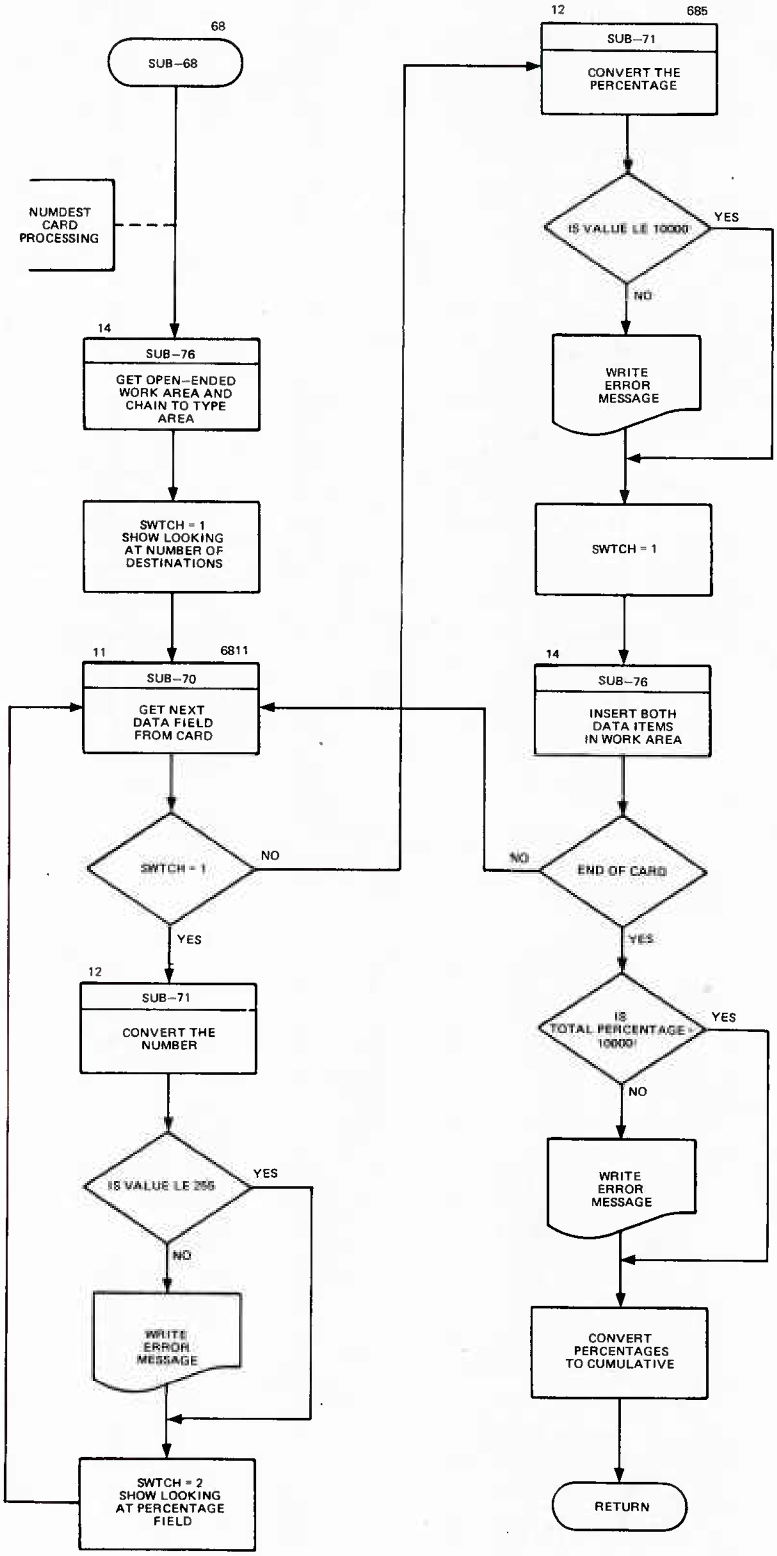
Statistical Message Generator Program (Sheet 6 of 17)



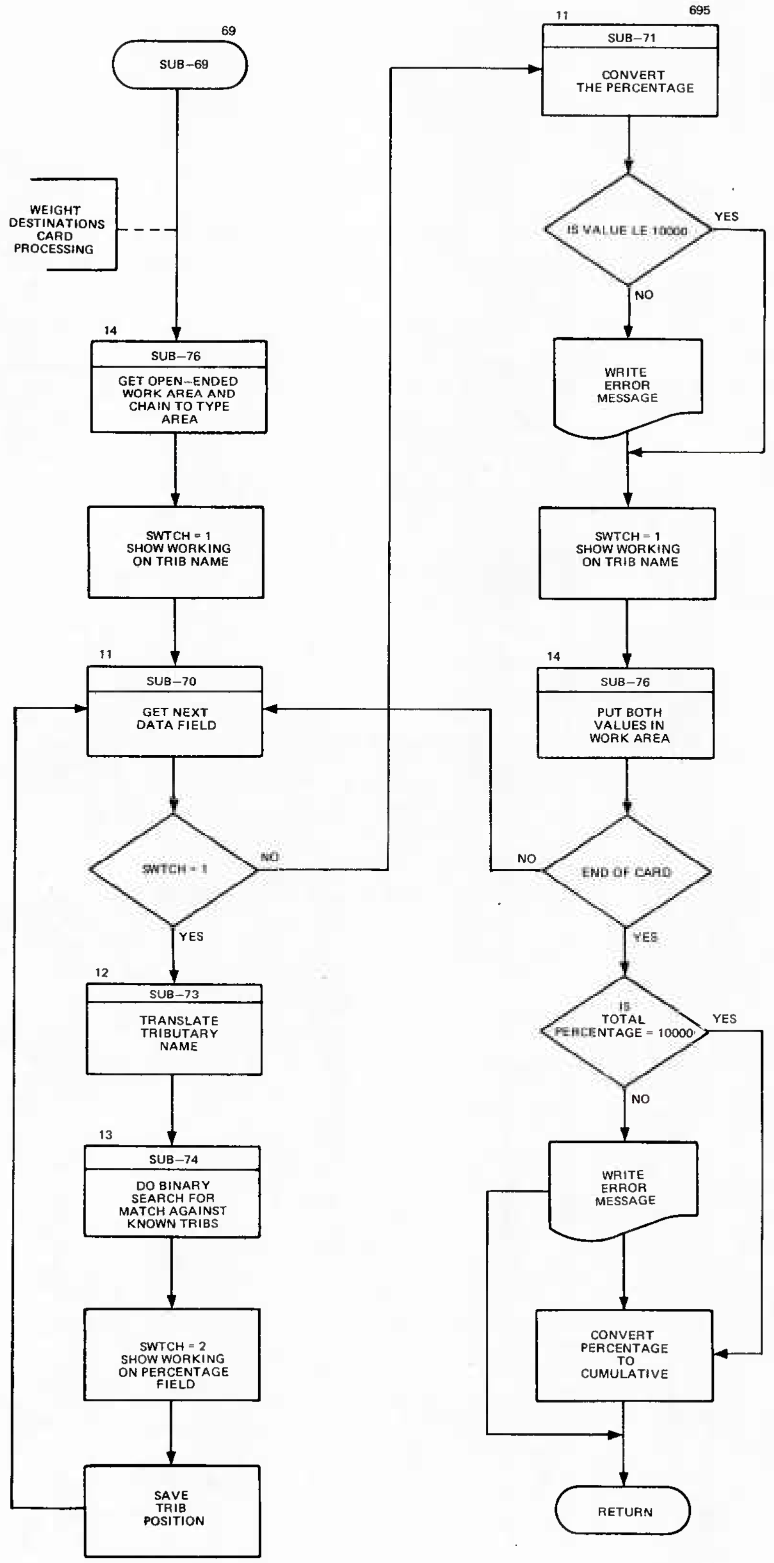
Statistical Message Generator Program (Sheet 7 of 17)


65
SUB–65
PRIORITY CARD OR SECURITY CARD PROCESSING
SET UP POINTERS IN TYPE AREA DEPENDING ON CARD TYPE
SWTCH = 1 SHOW LOOKING AT PRIORITY OR SECURITY CODE
11
650
SUB–70
GET NEXT FIELD ON CARD
IS SWITCH = 1
NO
YES
IS CODE OK
YES
NO
WRITE ERROR MESSAGE CODE = 0
SWTCH = 2 SHOW LOOKING AT PERCENTAGE
SAVE LOCATION IN TYPE AREA CORRESPONDING TO THIS CODE
12
655
SUB–71
CONVERT THE PERCENTAGE
IS VALUE LE 10000
YES
NO
WRITE ERROR MESSAGE
STORE PERCENTAGE INTO WORK AREA
NO
END OF CARD
YES
IS TOTAL PERCENTAGE = 10000
YES
NO
WRITE ERROR MESSAGE
IS THIS SECURITY CARD
NO
YES
CHANGE PERCENTAGES TO CUMULATIVE
RETURN

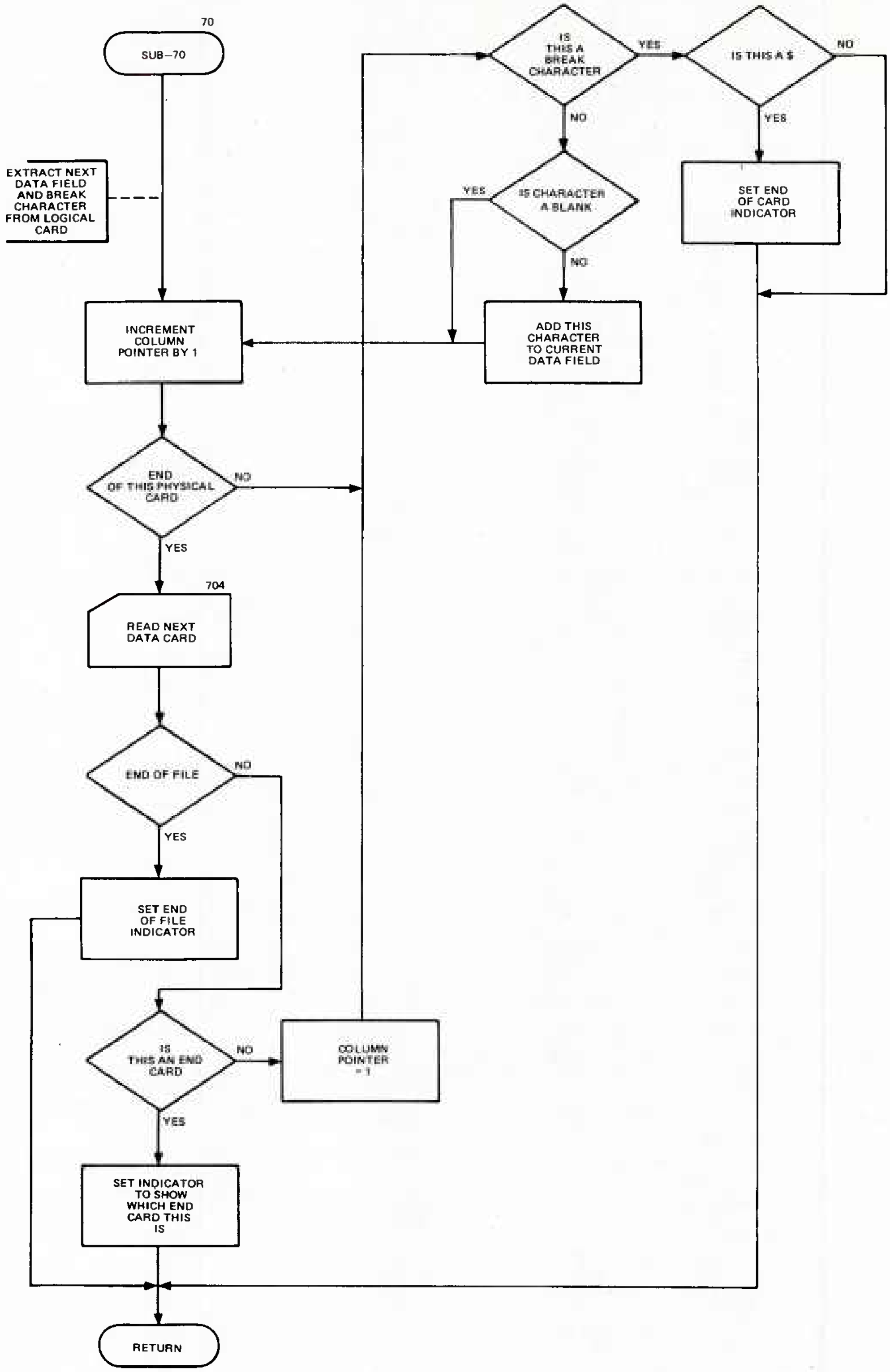
Statistical Message Generator Program (Sheet 8 of 17)



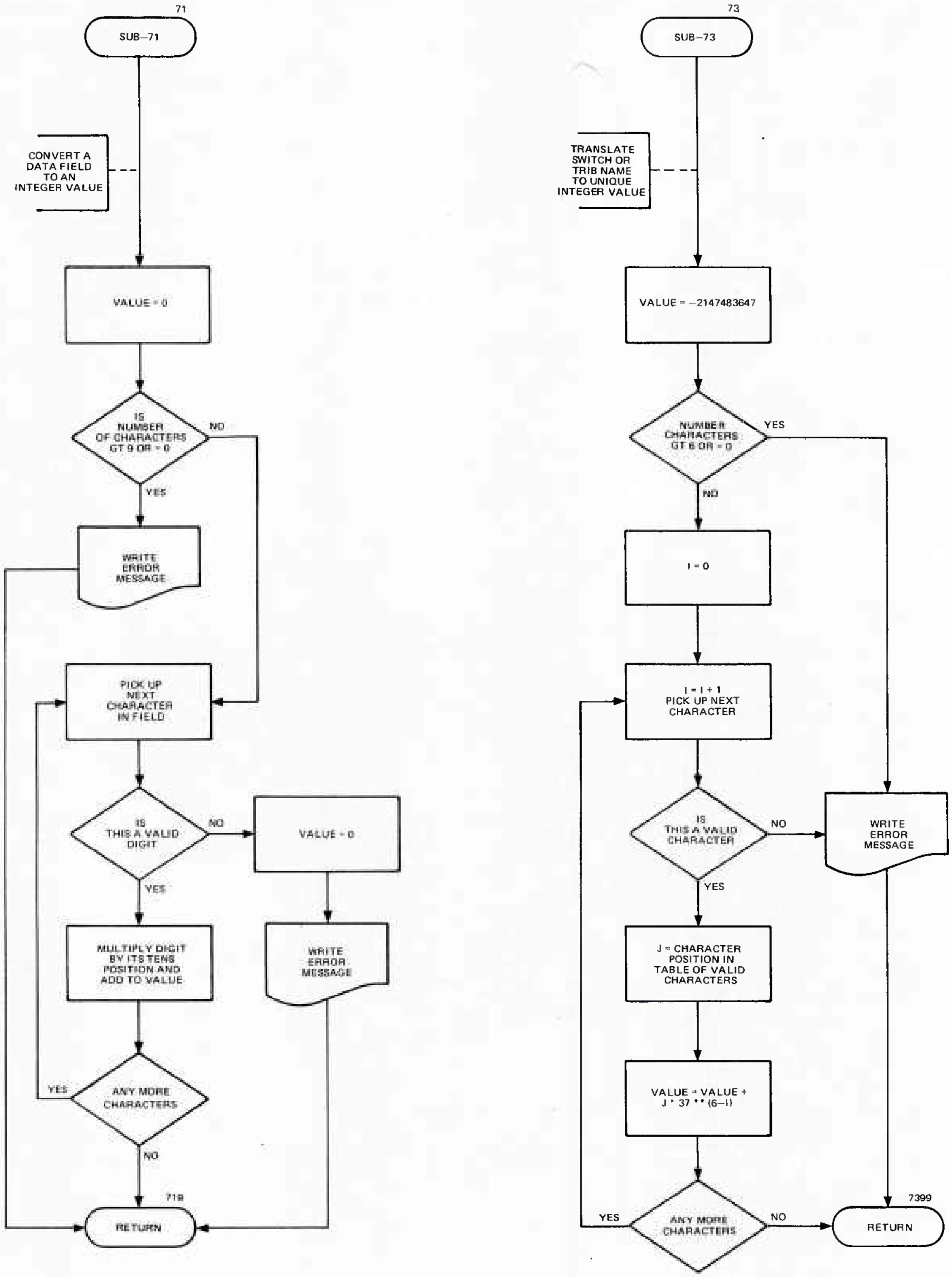
Statistical Message Generator Program (Sheet 9 of 17)


69
SUB–69
WEIGHT DESTINATIONS CARD PROCESSING
14
SUB–76
GET OPEN–ENDED WORK AREA AND CHAIN TO TYPE AREA
SWTCH = 1 SHOW WORKING ON TRIB NAME
11
SUB–70
GET NEXT DATA FIELD
SWTCH = 1
NO
YES
12
SUB–73
TRANSLATE TRIBUTARY NAME
13
SUB–74
DO BINARY SEARCH FOR MATCH AGAINST KNOWN TRIBS
SWTCH = 2 SHOW WORKING ON PERCENTAGE FIELD
SAVE TRIB POSITION
11
695
SUB–71
CONVERT THE PERCENTAGE
IS VALUE LE 10000
YES
NO
WRITE ERROR MESSAGE
SWTCH = 1 SHOW WORKING ON TRIB NAME
14
SUB–76
PUT BOTH VALUES IN WORK AREA
NO
END OF CARD
YES
IS TOTAL PERCENTAGE = 10000
YES
NO
WRITE ERROR MESSAGE
CONVERT PERCENTAGE TO CUMULATIVE
RETURN

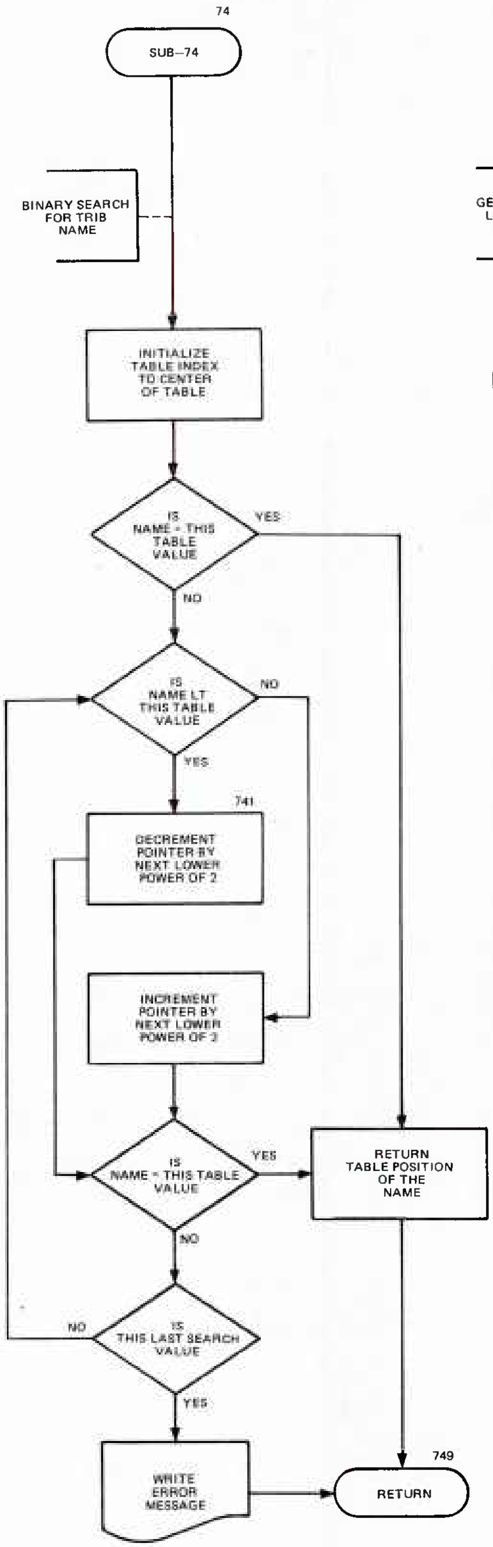
Statistical Message Generator Program (Sheet 10 of 17)

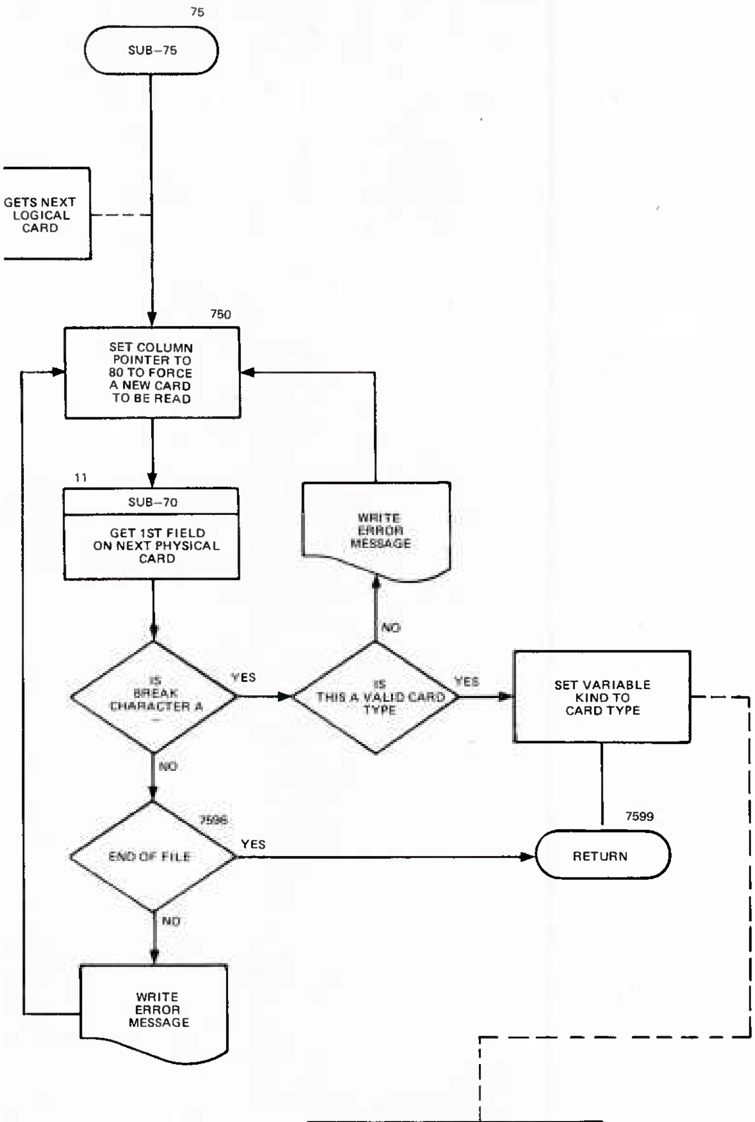

70
SUB–70
EXTRACT NEXT DATA FIELD AND BREAK CHARACTER FROM LOGICAL CARD
INCREMENT COLUMN POINTER BY 1
END OF THIS PHYSICAL CARD
NO
YES
704
READ NEXT DATA CARD
END OF FILE
NO
YES
SET END OF FILE INDICATOR
IS THIS AN END CARD
NO
YES
COLUMN POINTER = 1
SET INDICATOR TO SHOW WHICH END CARD THIS IS
RETURN
IS THIS A BREAK CHARACTER
YES
NO
IS CHARACTER A BLANK
YES
NO
ADD THIS CHARACTER TO CURRENT DATA FIELD
IS THIS A $
NO
YES
SET END OF CARD INDICATOR

Statistical Message Generator Program (Sheet 11 of 17)


71
SUB–71
CONVERT A DATA FIELD TO AN INTEGER VALUE
VALUE = 0
IS NUMBER OF CHARACTERS GT 9 OR = 0
NO
YES
WRITE ERROR MESSAGE
PICK UP NEXT CHARACTER IN FIELD
IS THIS A VALID DIGIT
NO
VALUE = 0
WRITE ERROR MESSAGE
YES
MULTIPLY DIGIT BY ITS TENS POSITION AND ADD TO VALUE
YES
ANY MORE CHARACTERS
NO
719
RETURN
73
SUB–73
TRANSLATE SWITCH OR TRIB NAME TO UNIQUE INTEGER VALUE
VALUE = –2147483647
NUMBER CHARACTERS GT 6 OR = 0
YES
NO
I = 0
I = I + 1 PICK UP NEXT CHARACTER
IS THIS A VALID CHARACTER
NO
WRITE ERROR MESSAGE
YES
J = CHARACTER POSITION IN TABLE OF VALID CHARACTERS
VALUE = VALUE + J * 37 ** (6–I)
YES
ANY MORE CHARACTERS
NO
7399
RETURN

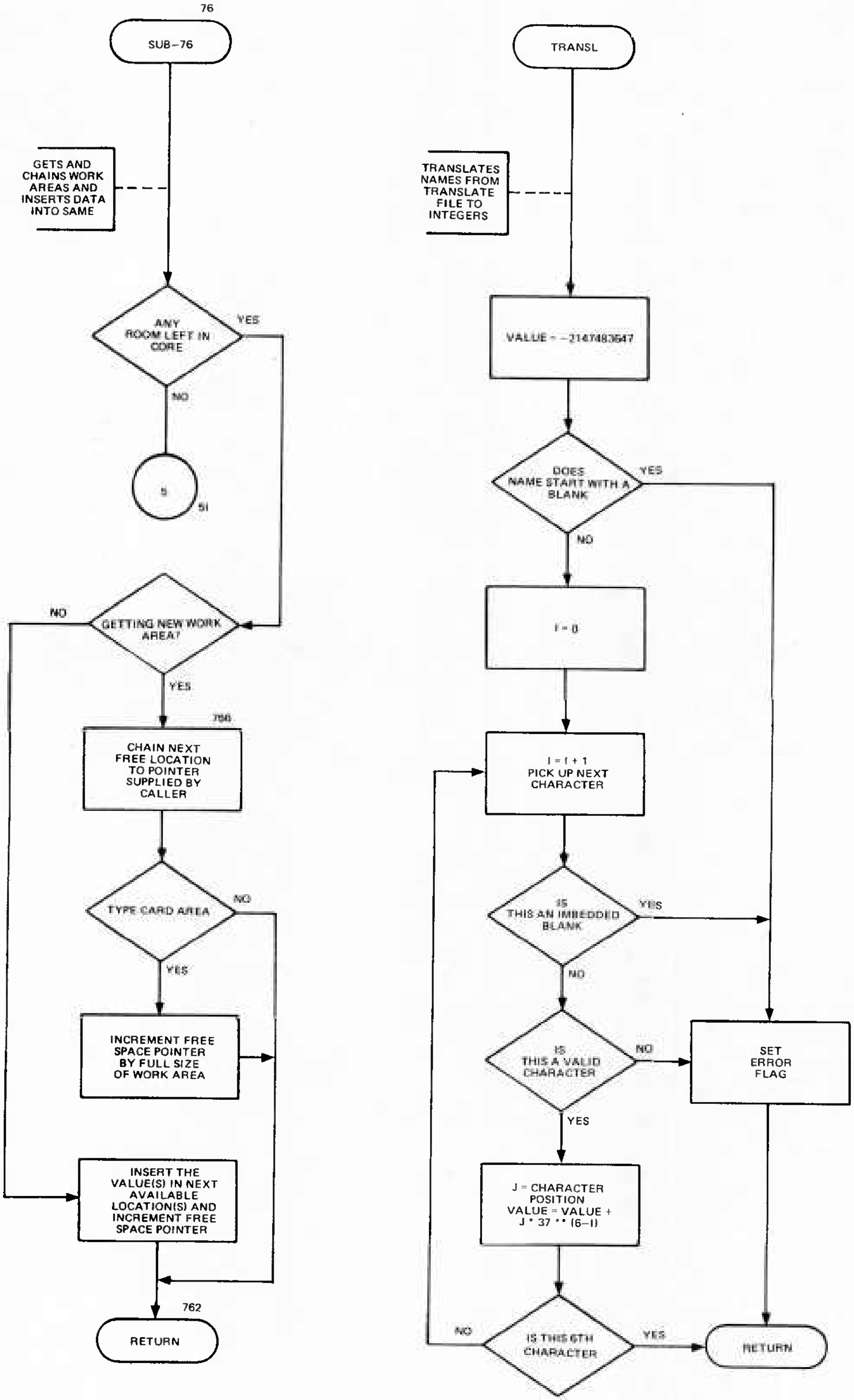
Statistical Message Generator Program (Sheet 12 of 17)


74
SUB–74
BINARY SEARCH FOR TRIB NAME
INITIALIZE TABLE INDEX TO CENTER OF TABLE
IS NAME = THIS TABLE VALUE
YES
NO
IS NAME LT THIS TABLE VALUE
NO
YES
741
DECREMENT POINTER BY NEXT LOWER POWER OF 2
INCREMENT POINTER BY NEXT LOWER POWER OF 2
IS NAME = THIS TABLE VALUE
YES
RETURN TABLE POSITION OF THE NAME
NO
IS THIS LAST SEARCH VALUE
NO
YES
WRITE ERROR MESSAGE
749
RETURN

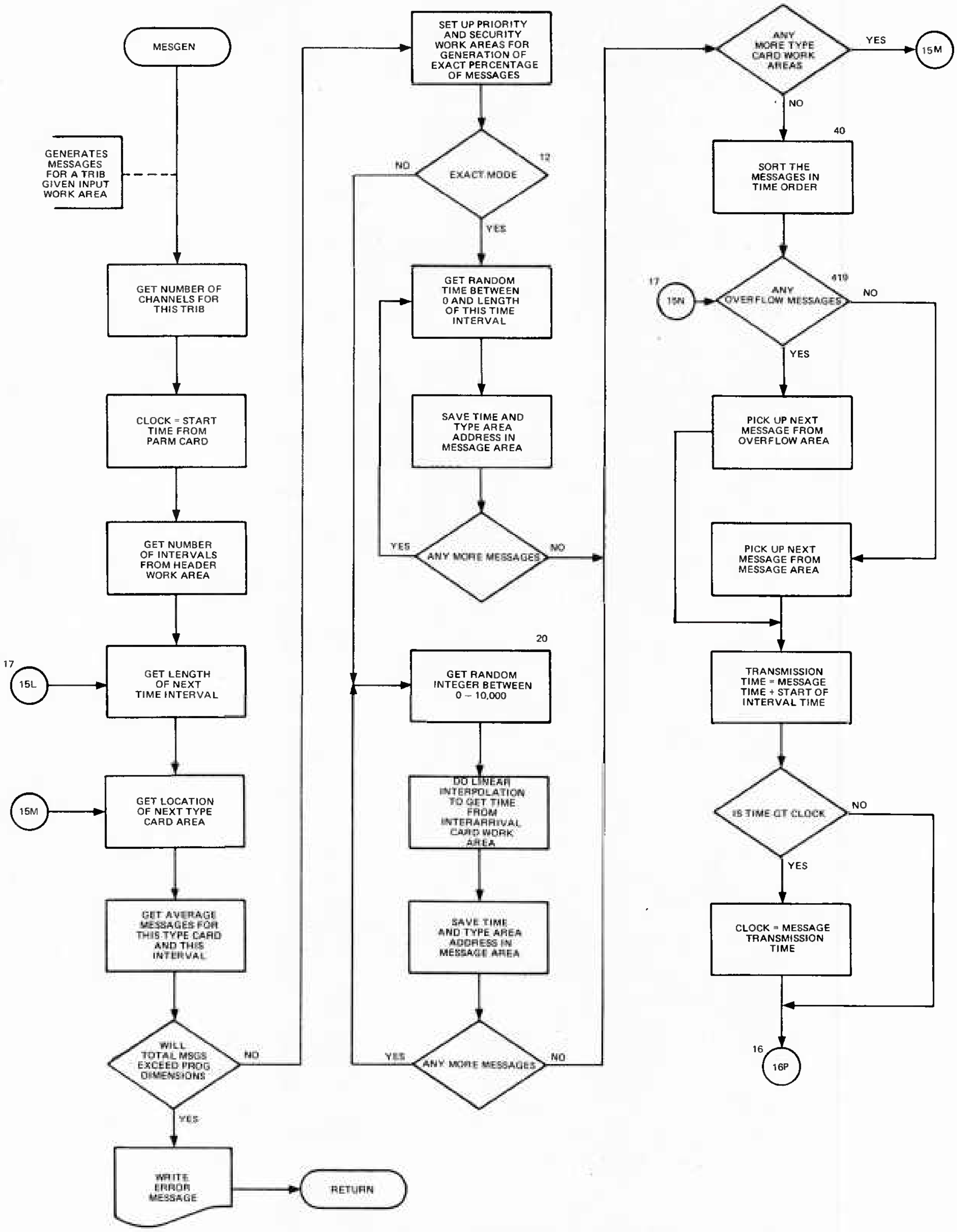

75
SUB–75
GETS NEXT LOGICAL CARD
750
SET COLUMN POINTER TO 80 TO FORCE A NEW CARD TO BE READ
11
SUB–70
GET 1ST FIELD ON NEXT PHYSICAL CARD
WRITE ERROR MESSAGE
NO
IS BREAK CHARACTER A —
YES
IS THIS A VALID CARD TYPE
YES
SET VARIABLE KIND TO CARD TYPE
NO
7596
END OF FILE
YES
7599
RETURN
NO
WRITE ERROR MESSAGE

CARD TYPE	KIND =
TYPE	1
INTERVAL	2
AVERAGE MESSAGES	3
INTER ARRIVAL	4
PRIORITY	5
LENGTH	6
SECURITY	7
NUMDEST	8
WEIGHT DESTINATIONS	9

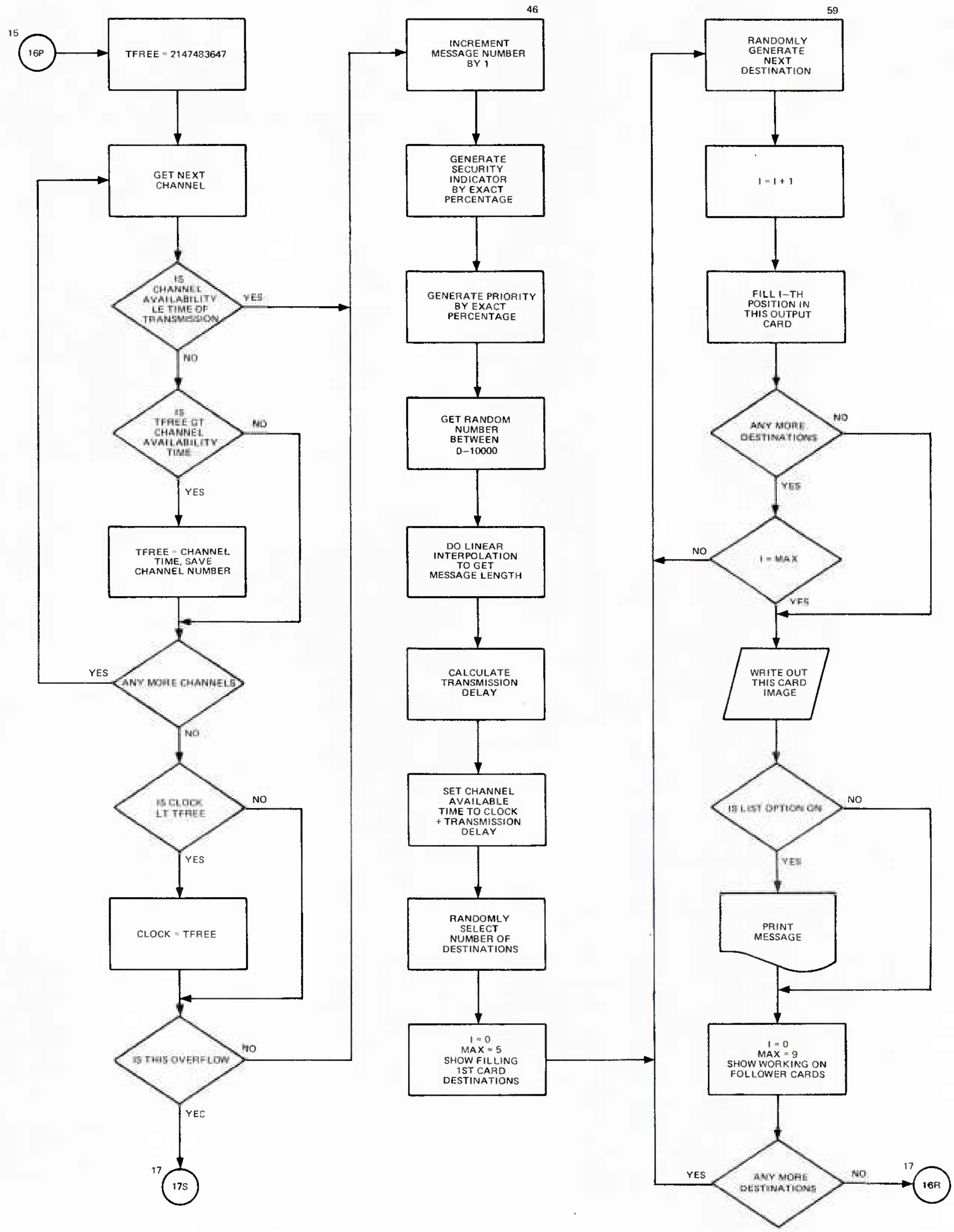
Statistical Message Generator Program (Sheet 13 of 17)


76
SUB-76
GETS AND CHAINS WORK AREAS AND INSERTS DATA INTO SAME
ANY ROOM LEFT IN CORE
YES
NO
5
51
NO
GETTING NEW WORK AREA?
YES
766
CHAIN NEXT FREE LOCATION TO POINTER SUPPLIED BY CALLER
TYPE CARD AREA
NO
YES
INCREMENT FREE SPACE POINTER BY FULL SIZE OF WORK AREA
INSERT THE VALUE(S) IN NEXT AVAILABLE LOCATION(S) AND INCREMENT FREE SPACE POINTER
762
RETURN
TRANSL
TRANSLATES NAMES FROM TRANSLATE FILE TO INTEGERS
VALUE = -2147483647
DOES NAME START WITH A BLANK
YES
NO
I = 0
I = I + 1
PICK UP NEXT CHARACTER
IS THIS AN IMBEDDED BLANK
YES
NO
IS THIS A VALID CHARACTER
NO
YES
SET ERROR FLAG
J = CHARACTER POSITION
VALUE = VALUE + J * 37 ** (6-I)
NO
IS THIS 6TH CHARACTER
YES
RETURN

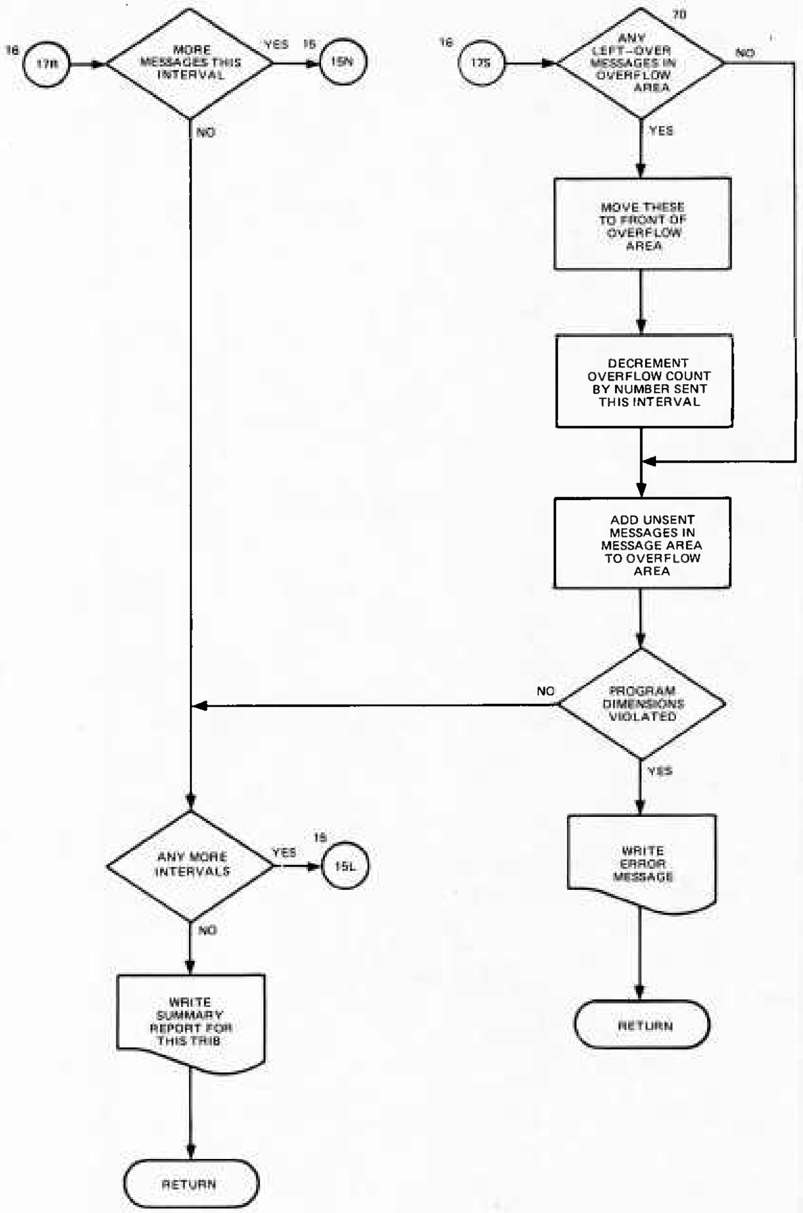
Statistical Message Generator Program (Sheet 14 of 17)


MESGEN
GENERATES MESSAGES FOR A TRIB GIVEN INPUT WORK AREA
GET NUMBER OF CHANNELS FOR THIS TRIB
CLOCK = START TIME FROM PARM CARD
GET NUMBER OF INTERVALS FROM HEADER WORK AREA
17
15L
GET LENGTH OF NEXT TIME INTERVAL
15M
GET LOCATION OF NEXT TYPE CARD AREA
GET AVERAGE MESSAGES FOR THIS TYPE CARD AND THIS INTERVAL
WILL TOTAL MSGS EXCEED PROG DIMENSIONS
NO
YES
WRITE ERROR MESSAGE
RETURN
SET UP PRIORITY AND SECURITY WORK AREAS FOR GENERATION OF EXACT PERCENTAGE OF MESSAGES
12
EXACT MODE
NO
YES
GET RANDOM TIME BETWEEN 0 AND LENGTH OF THIS TIME INTERVAL
SAVE TIME AND TYPE AREA ADDRESS IN MESSAGE AREA
ANY MORE MESSAGES
YES
NO
20
GET RANDOM INTEGER BETWEEN 0 – 10,000
DO LINEAR INTERPOLATION TO GET TIME FROM INTERARRIVAL CARD WORK AREA
SAVE TIME AND TYPE AREA ADDRESS IN MESSAGE AREA
ANY MORE MESSAGES
YES
NO
ANY MORE TYPE CARD WORK AREAS
YES
15M
NO
40
SORT THE MESSAGES IN TIME ORDER
17
15N
419
ANY OVERFLOW MESSAGES
NO
YES
PICK UP NEXT MESSAGE FROM OVERFLOW AREA
PICK UP NEXT MESSAGE FROM MESSAGE AREA
TRANSMISSION TIME = MESSAGE TIME + START OF INTERVAL TIME
IS TIME GT CLOCK
NO
YES
CLOCK = MESSAGE TRANSMISSION TIME
16
16P

Statistical Message Generator Program (Sheet 15 of 17)


15
16P
TFREE = 2147483647
GET NEXT CHANNEL
IS CHANNEL AVAILABILITY LE TIME OF TRANSMISSION
YES
NO
IS TFREE GT CHANNEL AVAILABILITY TIME
NO
YES
TFREE = CHANNEL TIME, SAVE CHANNEL NUMBER
ANY MORE CHANNELS
YES
NO
IS CLOCK LT TFREE
NO
YES
CLOCK = TFREE
IS THIS OVERFLOW
NO
YES
17
17S
46
INCREMENT MESSAGE NUMBER BY 1
GENERATE SECURITY INDICATOR BY EXACT PERCENTAGE
GENERATE PRIORITY BY EXACT PERCENTAGE
GET RANDOM NUMBER BETWEEN 0–10000
DO LINEAR INTERPOLATION TO GET MESSAGE LENGTH
CALCULATE TRANSMISSION DELAY
SET CHANNEL AVAILABLE TIME TO CLOCK + TRANSMISSION DELAY
RANDOMLY SELECT NUMBER OF DESTINATIONS
I = 0
MAX = 5
SHOW FILLING 1ST CARD DESTINATIONS
59
RANDOMLY GENERATE NEXT DESTINATION
I = I + 1
FILL I–TH POSITION IN THIS OUTPUT CARD
ANY MORE DESTINATIONS
NO
YES
I = MAX
NO
YES
WRITE OUT THIS CARD IMAGE
IS LIST OPTION ON
NO
YES
PRINT MESSAGE
I = 0
MAX = 9
SHOW WORKING ON FOLLOWER CARDS
ANY MORE DESTINATIONS
YES
NO
17
16R

Statisitcal Message Generator Program (Sheet 16 of 17)


16
17R
MORE MESSAGES THIS INTERVAL
YES
15
15N
NO
16
17S
70
ANY LEFT-OVER MESSAGES IN OVERFLOW AREA
NO
YES
MOVE THESE TO FRONT OF OVERFLOW AREA
DECREMENT OVERFLOW COUNT BY NUMBER SENT THIS INTERVAL
ADD UNSENT MESSAGES IN MESSAGE AREA TO OVERFLOW AREA
NO
PROGRAM DIMENSIONS VIOLATED
YES
WRITE ERROR MESSAGE
RETURN
ANY MORE INTERVALS
YES
15
15L
NO
WRITE SUMMARY REPORT FOR THIS TRIB
RETURN

Statistical Message Generator Program (Sheet 17 of 17)

SECTION 5 - MESSAGE SORT PROGRAM

5.1 PROGRAM REQUIREMENT DESCRIPTION

5.1.1 Program Name: Message Sort Program

5.1.2 Program Identification: DNMSGST1

5.1.3 Purpose

The Message Sort program sorts messages and events into chronological order for input to the Basic Simulation program. If more than one message file was created by previous programs, they are merged by the sort routine. Tributary names appearing in the message input files are translated to Status Table addresses as required by the following program. Special Events are an optional input to the program.

5.1.4 Requirements

The program is written in ANSI COBOL and requires a minimum of 200K bytes of memory for execution on an IBM 360 System.

5.2 INPUT SPECIFICATIONS

Inputs to the program are two or more disk files and an optional set of Special Event cards.

5.2.1 Translation File

The Translation file contains a cross-reference between the Switch, Link, and Tributary names and the address identification of these items in their respective Status Tables. The file is used in the translation of message inputs and Special Events. Generation of this file is described in Paragraph 2.3.

5.2.2 Discrete Message File

The Message Output file from the Discrete Message Generator program is an optional input file. Creation of this file is described in Paragraph 3.3.

5.2.3 Statistical Message File

The Message Output file from the Statistical Message Generator program is an optional input file. Creation of this file is described in Paragraph 4.3.

5.2.4 Special Events

The exogenous Special Events to the DIN Simulator are an optional input to the Message Sort program in the form of a punched card file. The format of these cards is given in Tables 5-1 through 5-11.

5.3 OUTPUT SPECIFICATIONS

The program generates one output file and a listing of special events when they are used.

5.3.1 Event-Message Output File

The output file from the Message Sort program is a merged file of messages and Special Events sorted into chronological order. The format for the output records is given in Tables 5-12 through 5-21.

5.3.2 Output Listing

The program output listing is a reproduction of the Special Events input file. The listing contains any error diagnostics generated.

5.4 PROGRAM LOGIC

5.4.1 General Discussion

The Message Sort source language program is built around two sort procedures imbedded in the COBOL source code. All file processing, data manipulation and name-to-address translation is performed within the Input and Output procedures for the two sorts.

The Input procedure for the first sort reads in and processes the Translation file which contains component (switch), link, and tributary records. When component or link records are read, the names and addresses necessary for the name-to-address translation are written into arrays for later reference. When tributary records are encountered in the Translation file they are written in their format onto a sort work file.

The end of the Translation file signals the end of the above Input procedure. The first sort is then executed. This sorts the tributary records into ascending order of tributary name. The Output procedure of this sort writes the tributary names and addresses into arrays for the name-to-address translation described below. The program then initializes several pointers, an increment table, and the unused area of the tributary name array to facilitate the later use of a binary-search procedure to translate tributary names. This completes the Output procedure of the first sort procedure.

Control is next passed to the Input procedure for the second sort. This procedure performs the main processing of Message and Special Events files. If Discrete and/or Statistical Message files are present, the files are processed identically, Statistical file first, in these five general steps:

1. Read a message record; if end of file is reached go to Step 5
2. Translate origin name and all destination names to Status Table addresses using binary-search routine
3. Write "translated" record onto a sort work file
4. Return to Step 1
5. Close Message file and begin processing next Message file (if one) or Special Events file.

The Special Events file is then processed by the same sort Input procedure in the following five general steps:

1. Read a Special Events record. If end of file is reached go to Step 5

2. Rearrange record into EMI-Special Events format (Tables 5-12 through 5-21) translating component, link, and tributary names to Status Table addresses using the three corresponding search routines (see Paragraph 5.4.2)

3. Write the "translated" record onto a sort work file

4. Return to Step 1

5. If syntax, logic, or program errors occurred during file processing, stop the run; otherwise, exit from this procedure to the Sort procedure.

The file containing all translated message and special events is next sorted into ascending order of the following sort keys (major keys first):

1. Event Time
2. Event Type
3. Message Number
4. Record Continuation Number (for multiple record message events).

The output of this sort is the main program output (the Event Message Output file).

5.4.2 Search Techniques

Three search routines are used in the Message Sort program for translating component, link, and tributary names to Status Table addresses. Two of these routines are virtually identical and perform straight linear searches of component and link name arrays, because of the relatively small sizes of these arrays.

The tributary name array tends to be large (up to 3000 entries) and linear searching would be time consuming; therefore, a binary search technique is utilized. The search technique consists of successive division of the array into halves and comparing the current search argument to the middle array entry of each division. If the array is sorted into ascending or descending order on the field to be searched, this technique guarantees to find each search argument, or state definintely that it is not in the search array, in relatively few steps. The technique works especially well for arrays which are of length 2^N-1 where N is some positive integer; and an array of size 2^N-1 can be searched in a maximum of N steps for each search argument.

When the number of array entries to be searched is not exactly 2^N-1, it is common practice to dimension the search array to the lowest value of 2^N-1 which is greater than or equal to the number of array entries. The unused array entries are then filled with some constant value which gives a predictable result for any comparison with that value.

Specifically, the tributary name array of the Message Sort program is dimensioned to 4095 (i.e., $2^{12}-1$). If a Translation table used by the program contains 3500 entries, locations 1 through 3500 of the name array are filled with the tributary names in ascending order of name. This is the reason for the first sort procedure discussed in Paragraph 5.4.1. Locations 3501 through 4095 are filled with the COBOL special named HIGH-VALUES, so that any search argument compares as less than any of the entries 3501 through 4095. This array can be searched in a maximum of N (=12) steps, and the average number of steps to find a search argument is approximately N-1 (=11). A linear search of the array would take, by comparison, a maximum of 3500 steps and an average of 1750 steps.

Flow charts of the program logic are included at the end of this section.

5.5 MACHINE DEPENDENCE

The Message Sort program is written in the American National Standards Institute (ANSI) COBOL language and therefore should execute on any system which supports this language.

Table 5-1. Special Events Message Input Card Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-8	Card Identification	Contains MESSAGE
9	Blank	
10-18	Time of Event	Event initiation time. May not exceed 36 hours. Seconds and milliseconds may be left blank.
10-11	Hours	
12-13	Minutes	
14-15	Seconds	
16-18	Milliseconds	
19	Blank	
20-25	Message Number	Identification of message. Any unique number from 1 to 999999 (right justified).
26	Blank	
27-32	Origination Tributary Name	Model name for tributary at which message originates.
33	Blank	
34-37	Message Length	Expressed in tenths of line blocks. Maximum value is 5110 (i.e., 511 line blocks).
38	Blank	

39	Message Type	T = Teletype
		C = Punched Card
		D = Data

40	Blank	

41	Message Priority	W = Flash Override
		Z = Flash
		O = Operational Immediate
		P = Priority
		R = Routine

42	Blank	

43	Message Security	A = Special
		T = Top Secret
		S = Secret
		C = Confidential
		U = Unclassified

44	Blank	
45-50	Destination No. 1	Name of 1st Destination Tributary (left justified).

Table 5-1. Special Events Message Input Card Format (Continued)

CARD COLUMN	DESCRIPTION	EXPLANATION
51	Blank	
52-57	Destination No. 2	Name of 2nd Destination Tributary (left-justified).
58	Blank	
59-64	Destination No. 3	Name of 3rd Destination Tributary (left-justified).
65	Blank	
66-71	Destination No. 4	Name of 4th Destination Tributary (left-justified).
72	Blank	
73-78	Destination No. 5	Name of 5th Destination Tributary (left-justified).
79-80	Blank	

Table 5-2. Special Events Message Continuation Input Card Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-8	Card Identification	Contains MSG CON*
9	Blank	
10-15	Message Number	Must be the same as preceding message input card.
16	Blank	
17-22	6th, 15th, etc. Destination	Destination Tributary name (left justified)
23	Blank	
24-29	7th, 16th, etc. Destination	Destination Tributary name (left justified)
30	Blank	
31-36	8th, 17th, etc. Destination	Destination Tributary name (left justified)
37	Blank	
38-43	9th, 18th, etc. Destination	Destination Tributary name (left justified)
44	Blank	
45-50	10th, 19th, etc. Destination	Destination Tributary name (left justified)
51	Blank	
52-57	11th, 20th, etc. Destination	Destination Tributary name (left justified)
58	Blank	
59-64	12th, 21st, etc. Destination	Destination Tributary name (left justified)
65	Blank	
66-71	13th, 22nd, etc. Destination	Destination Tributary name (left justified)
72	Blank	
73-78	14th, 23rd, etc. Destination	Destination Tributary name (left justified)
79-80	Blank	

*This card must follow its associated message card.

Table 5-3. Special Events Link or Tributary Outage Card Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-8	Identification	Contains LINK OUT
9	Blank	
10-18	Time of Event	Event initiation time. May not exceed 36 hours. Seconds and milliseconds may be left blank.
10-11	Hours	
12-13	Minutes	
14-15	Seconds	
16-18	Milliseconds	
19	Blank	
20-25	Link Name	Link or Tributary Name (left justified)
26	Blank	
27-32	Originating Switch	When a link outage is specified the originating switch must also be specified; otherwise this field may be blank.
33-35	Blank	
36-38	Channel Number	Channel number to be placed out of service. If zero or blank, all channels will be affected.
39	Altroute Indicator	When this field contains any character, the low priority altroute indicator is reversed (links only).
40	Blank	
41	Altroute Indicator	When this field contains any character, the low priority altroute indicator for the reverse direction of the link is reversed (links only).
42	Blank	
43	One Direction Indicator	When the field contains any character, the outage applies only to the primary direction of the link. The reverse direction is not affected (links only).
44	Blank	
45-53	Restoral Time	If a restoral time is specified, a restoral event of identical characteristics will be generated.
45-46	Hours	

Table 5-3. Special Events Link or Tributary Outage Card Format (Continued)

CARD COLUMN	DESCRIPTION	EXPLANATION
47-48	Minutes	
49-50	Seconds	
51-53	Milliseconds	
54-80	Blank	

Table 5-4. Special Events Link or Tributory Restoral Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-8	Identification	Contains LINK IN
9	Blank	
10-18	Time of Event	Event initiation time. May not exceed 36 hours. Seconds and milliseconds may be left blank.
10-11	Hours	
12-13	Minutes	
14-15	Seconds	
16-18	Milliseconds	
19	Blank	
20-25	Link Name	Link or Tributary Name (left justified).
26	Blank	
27-32	Originating Switch	When a link restoral is specified, the originating switch must also be specified; otherwise this field may be blank.
33-35	Blank	
36-38	Channel Number	Channel number to be returned to service. If zero or blank, all channels will be affected.
39	Altroute Indicator	When this field contains any character, the low priority altroute indicator is reversed (links only).
40	Blank	
41	Altroute Indicator	When this field contains any character, the low priority altroute indicator for the reverse direction of the link is reversed.
42	Blank	
43	One Direction Indicator	When the field contains any character, the restoral applies only to the primary direction of the link. The reverse direction is not affected.
44-80	Blank	

Table 5-5. Special Events Switch Outage Card Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-8	Card Identification	Contains SWT OUT
9	Blank	
10-18	Time of Event	Event initiation time. May not exceed 36 hours. Seconds and milliseconds may be left blank.
10-11	Hours	
12-13	Minutes	
14-15	Seconds	
16-18	Milliseconds	
19	Blank	
20-25	Switch Name	Name of Switch affected (left justified).
26	Blank	
27-35	Restoral Time	If a restoral time is specified, a restoral event of identical characteristics will also be generated.
27-28	Hours	
29-30	Minutes	
31-32	Seconds	
33-35	Milliseconds	
36-80	Blank	

Table 5-6. Special Events Switch Restoral Card Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-8	Card Identification	Contains SWT IN
9	Blank	
10-18	Time of Event	Event initiation time. May not exceed 36 hours. Seconds and milliseconds may be left blank.
10-11	Hours	
12-13	Minutes	
14-15	Seconds	
16-18	Milliseconds	
19	Blank	
20-25	Switch Name	Name of switch affected (left justified).
26-80	Blank	

Table 5-7. Special Events Change of Routing Input Card

CARD COLUMN	DESCRIPTION	EXPLANATION
1-8	Card Identification	Contains ROUTE
9	Blank	
10-18	Time Event	Event initiation time. May not exceed 36 hours. Seconds and milliseconds may be left blank.
10-11	Hours	
12-13	Minutes	
14-15	Seconds	
16-18	Milliseconds	
19	Blank	
20-25	Switch Name	Source switch name (left justified)
26	Blank	
27-32	Destination Switch Name	Destination switch name (left justified).
33	Blank	
34-39	Primary Link Name	Link Name (left justified). If blank, primary route will not be modified.
40	Blank	
41-46	Alternate Link Name	Link name (left justified). If blank, alternate route will not be modified.
47-80	Blank	

Table 5-8. Special Events Reversal of Altroute Indicator Card Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-8	Card Identification	Contains ALTROUTE
9	Blank	
10-18	Time of Event	Event initiation time. May not exceed 36 hours. Seconds and milliseconds may be left blank.
10-11	Hours	
12-13	Minutes	
14-15	Seconds	
16-18	Milliseconds	
19	Blank	
20-25	Link Name	Link name (left justified).
26	Blank	
27-32	Switch Name	Name of originating switch (left justified).
33-38	Blank	
39	Altroute Indicator	When this field contains any character, the low priority altroute indicator is reversed for the primary direction of the link.
40	Blank	
41	Altroute Indicator	When this field contains any character, the low priority altroute indicator is reversed for the reverse direction of the link.
42-80	Blank	

Table 5-9. Special Events Reversal of Preempts Indicator Card Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-8	Card Identification	Contains PREEMPT
9	Blank	
10-18	Time of Event	Event initiation time. May not exceed 36 hours. Seconds and milliseconds may be left blank
10-11	Hours	
12-13	Minutes	
14-15	Seconds	
16-18	Milliseconds	
19	Blank	
20-25	Link Name	Link or tributary name. When this event is executed, the indicator bit which allows Operational Immediate messages to preempt is reversed. (Bit is normally off).

Table 5-10. Special Events CARP Invocation Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-8	Card Identification	Contains CARP INV.
9	Blank	
10-18	Time of Event	Event initiation time. May not exceed 36 hours. Seconds and milliseconds may be left blank.
10-11	Hours	
12-13	Minutes	
14-15	Seconds	
16-18	Milliseconds	
19	Blank	
20-25	Original Tributary Name	Name of tributary whose messages are to be rerouted.
26	Blank	
27-32	Alternate Tributary Name	Name of tributary to whom messages are to be rerouted.
33-38	Blank	

39	Security Level	Classification of messages to be rerouted.
		A = Special
		T = Top Secret
		S = Secret
		C = Confidential
		U = Unclassified
		Blank = all messages

40-80	Blank	

Table 5-11. Special Events CARP Cancellation Card Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-8	Card Identification	Contains CARP CAN.
9	Blank	
10-18	Time of Event	Event initiation time. May not exceed 36 hours. Seconds and milliseconds may be left blank.
10-11	Hours	
12-13	Minutes	
14-15	Seconds	
16-18	Milliseconds	
19	Blank	
20-25	Original Tributary	See Table 8-10. Remaining fields in this card must be identical to the card which invoked CARP.
26	Blank	
27-32	Alternate Tributary Name	
33-38	Blank	
39	Security Level	
40-80	Blank	

Table 5-12. Event Message Record Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-2	Card sequence number	Indicates continuation cards for messages with numerous destinations
3-11	Time	Time of the event in milliseconds
12-13	Event Type	1 = Message Input 2 = Link Outage 3 = Link Restoral 4 = Switch Outage 5 = Switch Restoral 6 = Routing Change 7 = Reverse Altroute Indicator 8 = Reverse Preemption Indicator 9 = CARP
14	Blank	
15-20	Field 1	Defined for various Event Types
21-26	Field 2	
27-32	Field 3	
33-38	Field 4	
39-44	Field 5	
45-50	Field 6	
51-56	Field 7	
57-62	Field 8	
63-68	Field 9	
69-74	Field 10	
75-80	Field 11	

Table 5-13. Message Input Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-2	Card Sequence Number	01 for first card
3-11	Time	Time in milliseconds
12-13	Type	01 for Message Input
14	Blank	
15-20	Field 1	Message Number
21-26	Field 2	Originator
27-32	Field 3	Length
33-38	Field 4	Message Type
39-44	Field 5	Precedence .
45-50	Field 6	Security
51-56	Field 7	1st Destination
57-62	Field 8	2nd Destination
63-68	Field 9	3rd Destination
69-74	Field 10	4th Destination
75-80	Field 11	5th Destination

Table 5-14. Message Continuation Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-2	Card Sequence Number	02 or greater
3-11	Time	Time in milliseconds
12-13	Type	01 for Message Input
14	Blank	
15-20	Field 1	Message Number
21-26	Field 2	Originator
27-32	Field 3	6th, 15th, etc., Destination
33-38	Field 4	7th, 16th, etc.
39-44	Field 5	8th, 17th, etc.
45-50	Field 6	9th, 18th, etc.
51-56	Field 7	10th, 19th, etc.
57-62	Field 8	11th, 20th, etc.
63-68	Field 9	12th, 21st, etc.
69-74	Field 10	13th, 22nd, etc.
75-80	Field 11	14th, 23rd, etc.

Table 5-15. Link Outage Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-2	Card Sequence Number	01
3-11	TIME	Time in milliseconds
12-13	Event Type	02
14	Blank	
15-20	Field 1	Link or tributary address
21-26	Field 2	Altroute indicator primary direction (0 = no, 1 = yes)
27-32	Field 3	Altroute indicator reverse direction (0 = no, 1 = yes)
33-38	Field 4	Channel number affected (if 0, all channels)
39-44	Field 5	Direction (0 = both directions, 1 = primary direction only)

Table 5-16. Link Restoral Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-2	Card Sequence Number	01
3-11	TIME	Time in milliseconds
12-13	Event Type	03
14	Blank	
15-20	Field 1	Link or tributary address
21-26	Field 2	Altroute indicator primary direction (0 = no, 1 = yes)
27-32	Field 3	Altroute indicator reverse direction (0 = no, 1 = yes)
33-38	Field 4	Channel number affected (if 0, all channels)
39-44	Field 5	Direction (0 = both directions, 1 = primary direction only)

Table 5-17. Switch Outage Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-2	Card Sequence Number	01
3-11	TIME	Time in milliseconds
12-13	Event Type	04
14	Blank	
15-20	Field 1	Switch Number

Table 5-18. Switch Restoral Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-2	Card Sequence Number	01
3-11	TIME	Time in milliseconds
12-13	Event Type	05
14	Blank	
15-20	Field 1	Switch Number

Table 5-19. Routing Change Input Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-2	Card Sequence Number	01
3-11	TIME	Time in milliseconds
12-13	Event Type	06
14	Blank	
15-20	Field 1	Source Switch Number
21-26	Field 2	Destination Switch Number
27-32	Field 3	Primary Link Address
33-38	Field 4	Secondary Link Address

Table 5-20. Reverse Altroute Indicator Format

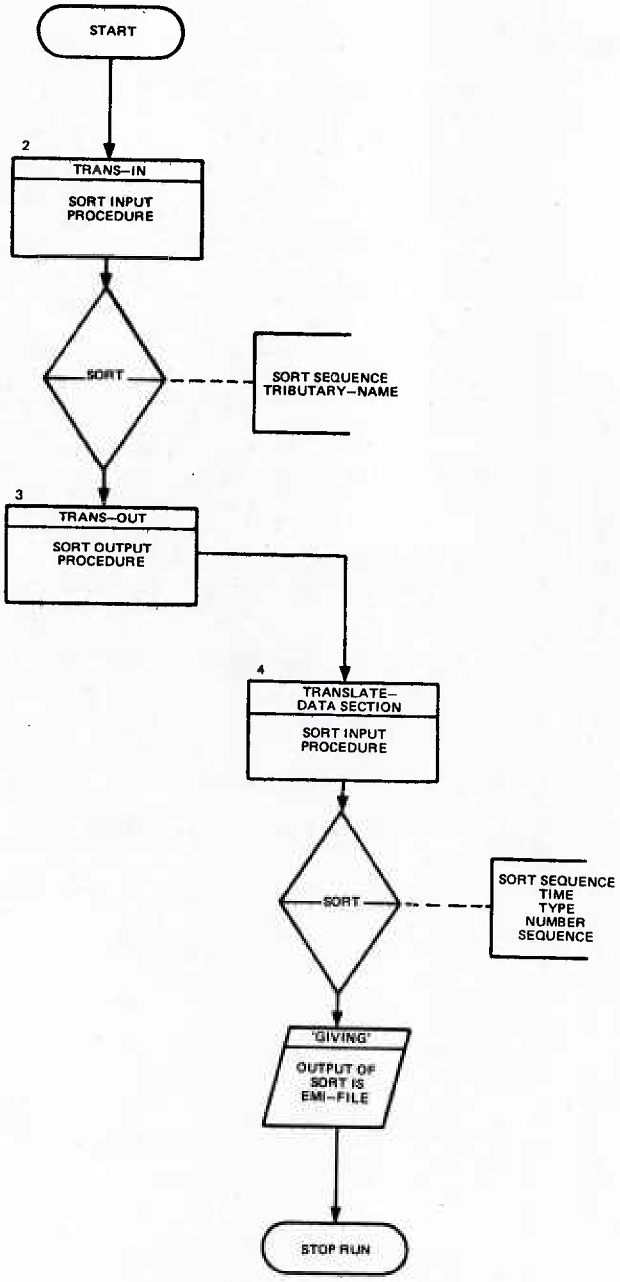
CARD COLUMN	DESCRIPTION	EXPLANATION
1-2	Card Sequence Number	01
3-11	TIME	Time in milliseconds
12-13	Event Type	07
14	Blank	
15-20	Field 1	Link Address
21-26	Field 2	Reversal of Primary direction (0 = no, 1 = yes)
27-32	Field 3	Reversal of Reverse direction (0 = no, 1 = yes)

Table 5-21. Reverse Preempt Indicator Format

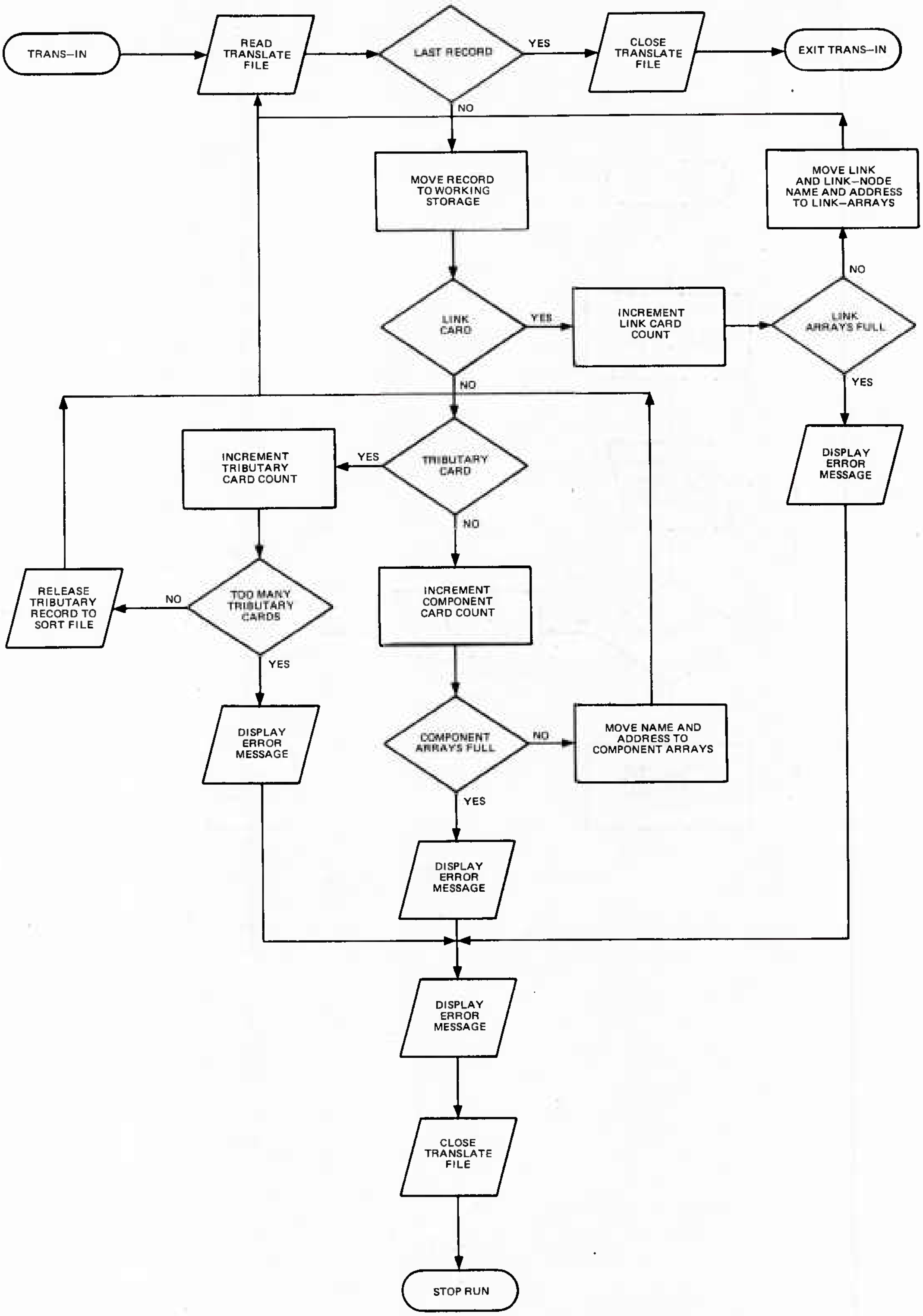
CARD COLUMN	DESCRIPTION	EXPLANATION
1-2	Card Sequence Number	01
3-11	TIME	Time in milliseconds
12-13	Event Type	08
14	Blank	
15-20	Field 1	Link or tributary address

Table 5-22 CARP Format

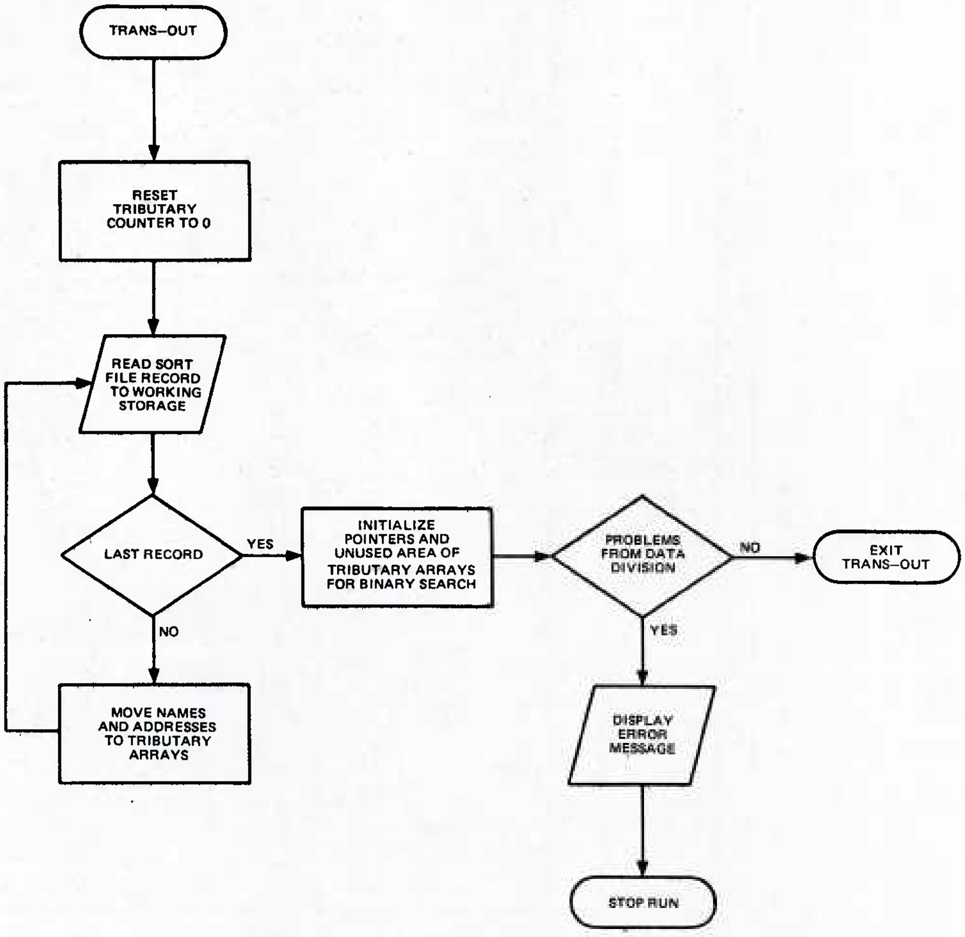
CARD COLUMN	DESCRIPTION	EXPLANATION
1-2	Card Sequence Number	01
3-11	TIME	Time in milliseconds
12-13	Event Type	09
14	Blank	
15-20	Field 1	Original Tributary
21-26	Field 2	Alternate Tributary
27-32	Field 3	Security level affected -1 All Levels 0 Unclassified only 1 Confidential only 2 Secret only 3 Top Secret only 4 Special only
33-38	Field 4	Type action 0 = Invocation 1 = Cancellation



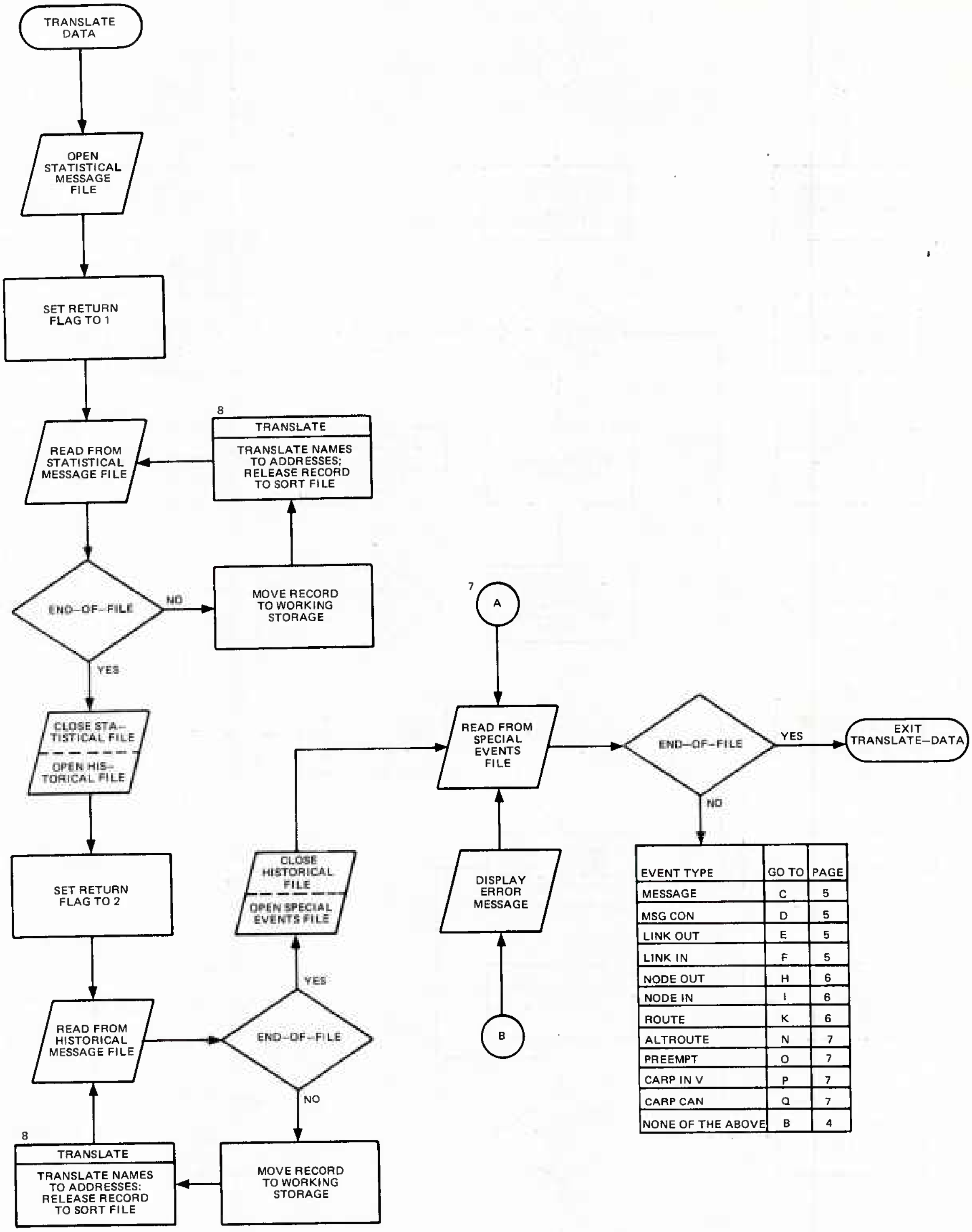
Message Sort Program (Sheet 1 of 9)



Message Sort Program (Sheet 2 of 9)

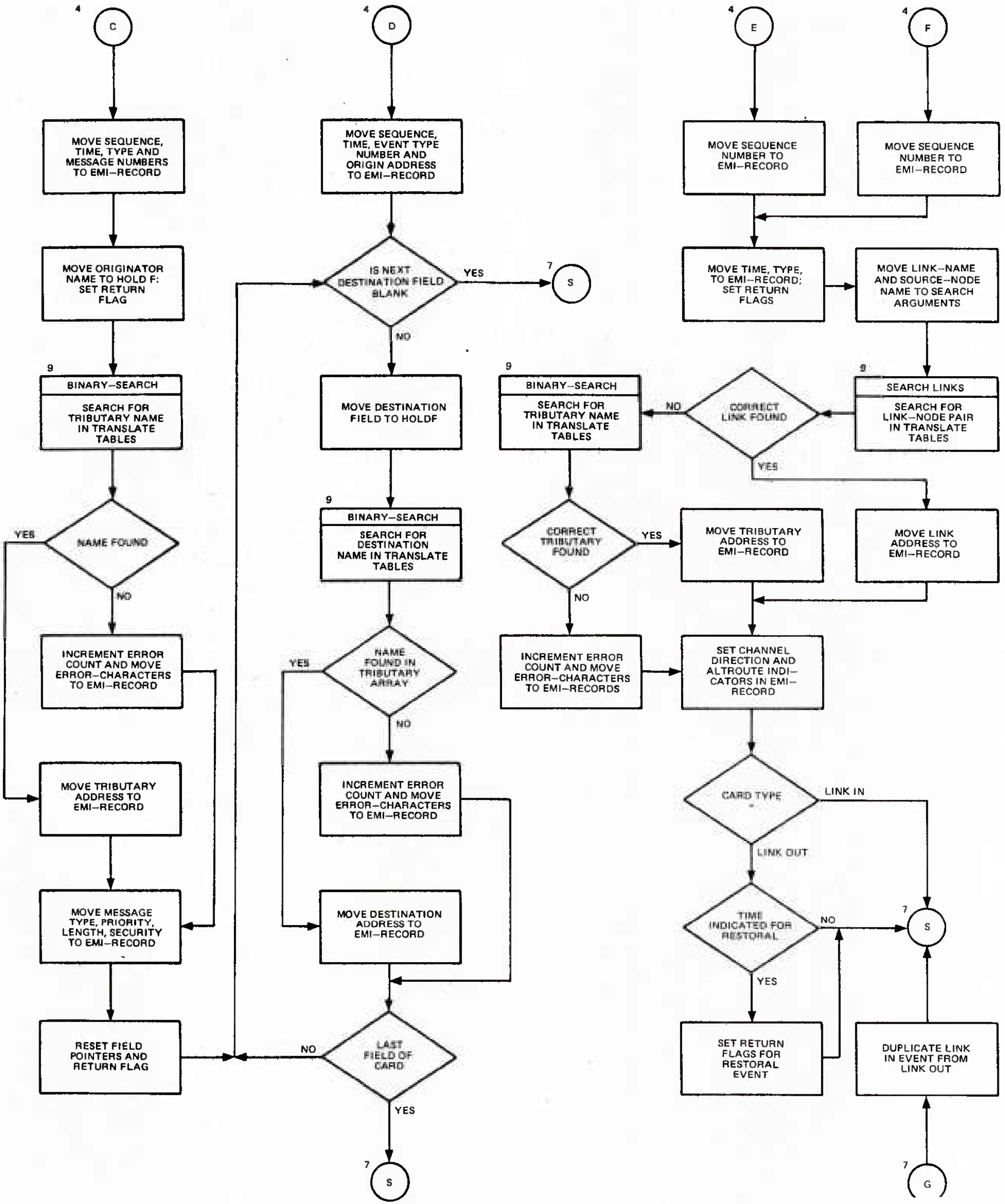

TRANS–OUT
RESET TRIBUTARY COUNTER TO 0
READ SORT FILE RECORD TO WORKING STORAGE
LAST RECORD
YES
NO
MOVE NAMES AND ADDRESSES TO TRIBUTARY ARRAYS
INITIALIZE POINTERS AND UNUSED AREA OF TRIBUTARY ARRAYS FOR BINARY SEARCH
PROBLEMS FROM DATA DIVISION
NO
EXIT TRANS–OUT
YES
DISPLAY ERROR MESSAGE
STOP RUN

Message Sort Program (Sheet 3 of 9)

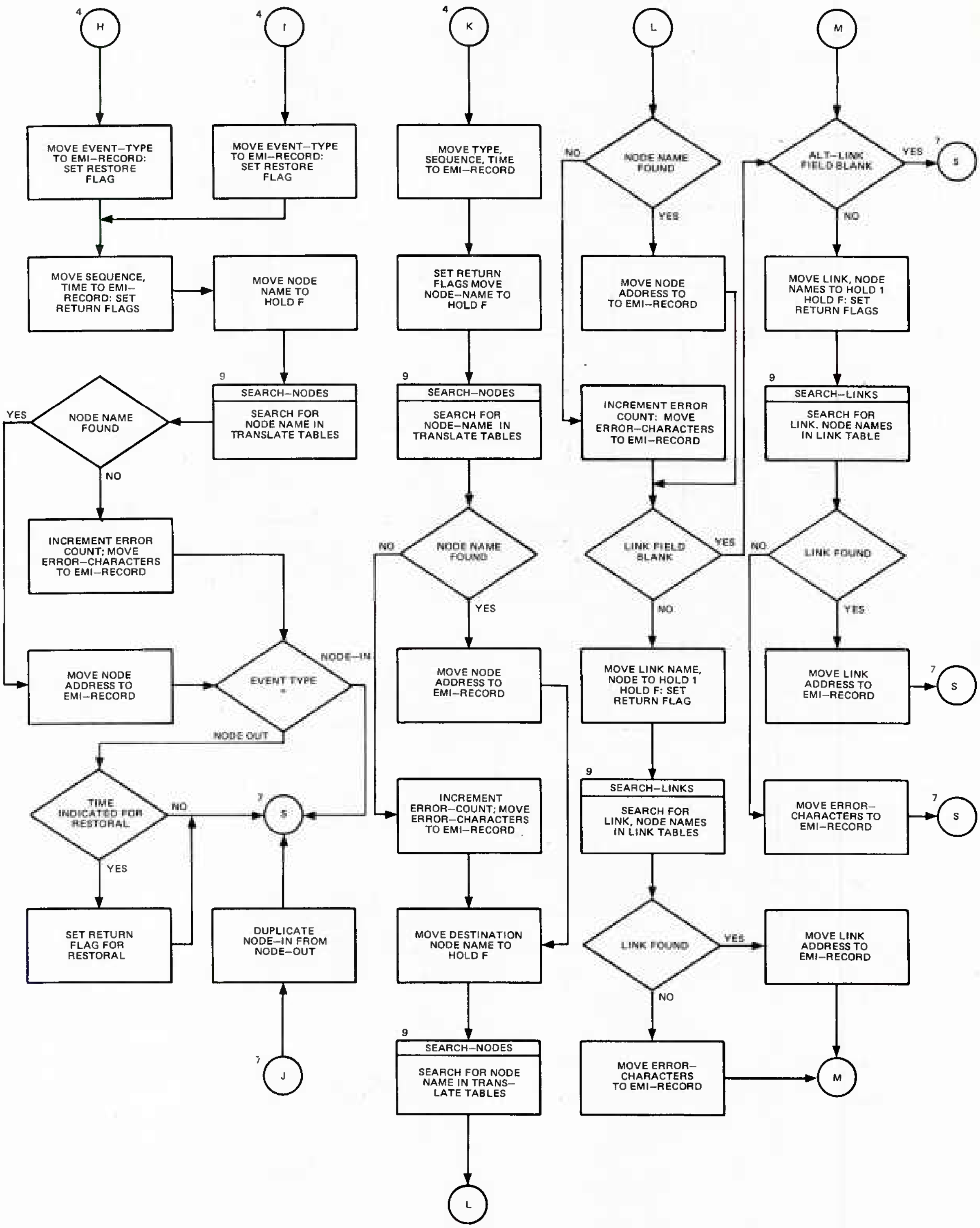

TRANSLATE DATA
OPEN STATISTICAL MESSAGE FILE
SET RETURN FLAG TO 1
READ FROM STATISTICAL MESSAGE FILE
8
TRANSLATE
TRANSLATE NAMES TO ADDRESSES; RELEASE RECORD TO SORT FILE
END-OF-FILE
NO
MOVE RECORD TO WORKING STORAGE
YES
CLOSE STATISTICAL FILE
OPEN HISTORICAL FILE
SET RETURN FLAG TO 2
READ FROM HISTORICAL MESSAGE FILE
END-OF-FILE
YES
NO
CLOSE HISTORICAL FILE
OPEN SPECIAL EVENTS FILE
MOVE RECORD TO WORKING STORAGE
8
TRANSLATE
TRANSLATE NAMES TO ADDRESSES: RELEASE RECORD TO SORT FILE
7
A
READ FROM SPECIAL EVENTS FILE
DISPLAY ERROR MESSAGE
B
END-OF-FILE
YES
EXIT TRANSLATE-DATA
NO

EVENT TYPE	GO TO	PAGE
MESSAGE	C	5
MSG CON	D	5
LINK OUT	E	5
LINK IN	F	5
NODE OUT	H	6
NODE IN	I	6
ROUTE	K	6
ALTROUTE	N	7
PREEMPT	O	7
CARP IN V	P	7
CARP CAN	Q	7
NONE OF THE ABOVE	B	4

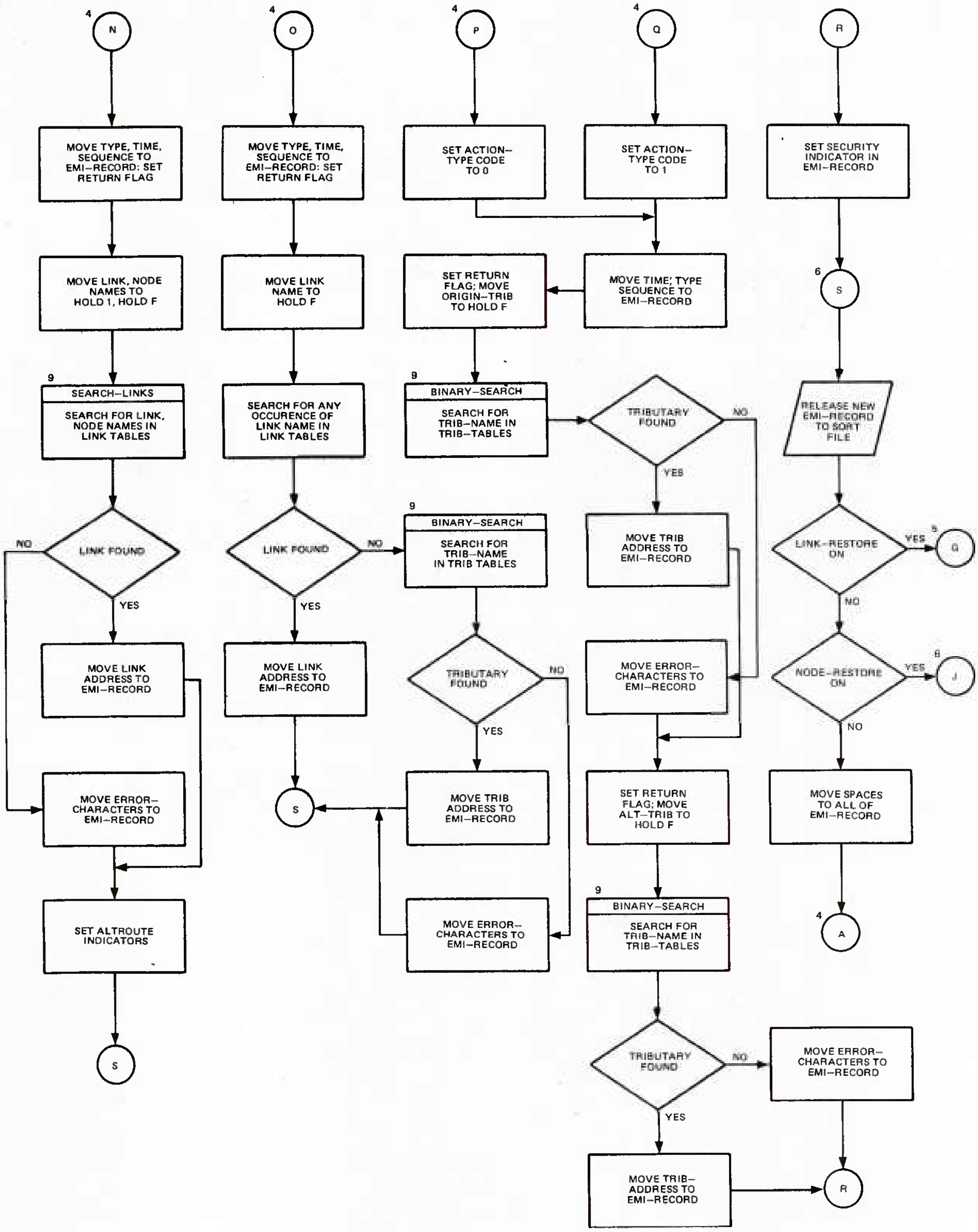
Message Sort Program (Sheet 4 of 9)


4
C
MOVE SEQUENCE, TIME, TYPE AND MESSAGE NUMBERS TO EMI–RECORD
MOVE ORIGINATOR NAME TO HOLD F: SET RETURN FLAG
9
BINARY–SEARCH
SEARCH FOR TRIBUTARY NAME IN TRANSLATE TABLES
NAME FOUND
YES
NO
INCREMENT ERROR COUNT AND MOVE ERROR–CHARACTERS TO EMI–RECORD
MOVE TRIBUTARY ADDRESS TO EMI–RECORD
MOVE MESSAGE TYPE, PRIORITY, LENGTH, SECURITY TO EMI–RECORD
RESET FIELD POINTERS AND RETURN FLAG
4
D
MOVE SEQUENCE, TIME, EVENT TYPE NUMBER AND ORIGIN ADDRESS TO EMI–RECORD
IS NEXT DESTINATION FIELD BLANK
YES
7
S
NO
MOVE DESTINATION FIELD TO HOLDF
9
BINARY–SEARCH
SEARCH FOR DESTINATION NAME IN TRANSLATE TABLES
NAME FOUND IN TRIBUTARY ARRAY
YES
NO
INCREMENT ERROR COUNT AND MOVE ERROR–CHARACTERS TO EMI–RECORD
MOVE DESTINATION ADDRESS TO EMI–RECORD
LAST FIELD OF CARD
NO
YES
7
S
4
E
MOVE SEQUENCE NUMBER TO EMI–RECORD
4
F
MOVE SEQUENCE NUMBER TO EMI–RECORD
MOVE TIME, TYPE, TO EMI–RECORD; SET RETURN FLAGS
MOVE LINK–NAME AND SOURCE–NODE NAME TO SEARCH ARGUMENTS
9
SEARCH LINKS
SEARCH FOR LINK–NODE PAIR IN TRANSLATE TABLES
CORRECT LINK FOUND
NO
YES
9
BINARY–SEARCH
SEARCH FOR TRIBUTARY NAME IN TRANSLATE TABLES
CORRECT TRIBUTARY FOUND
YES
NO
MOVE TRIBUTARY ADDRESS TO EMI–RECORD
MOVE LINK ADDRESS TO EMI–RECORD
INCREMENT ERROR COUNT AND MOVE ERROR–CHARACTERS TO EMI–RECORDS
SET CHANNEL DIRECTION AND ALTROUTE INDI– CATORS IN EMI– RECORD
CARD TYPE
LINK IN
LINK OUT
TIME INDICATED FOR RESTORAL
NO
YES
7
S
SET RETURN FLAGS FOR RESTORAL EVENT
DUPLICATE LINK IN EVENT FROM LINK OUT
7
G

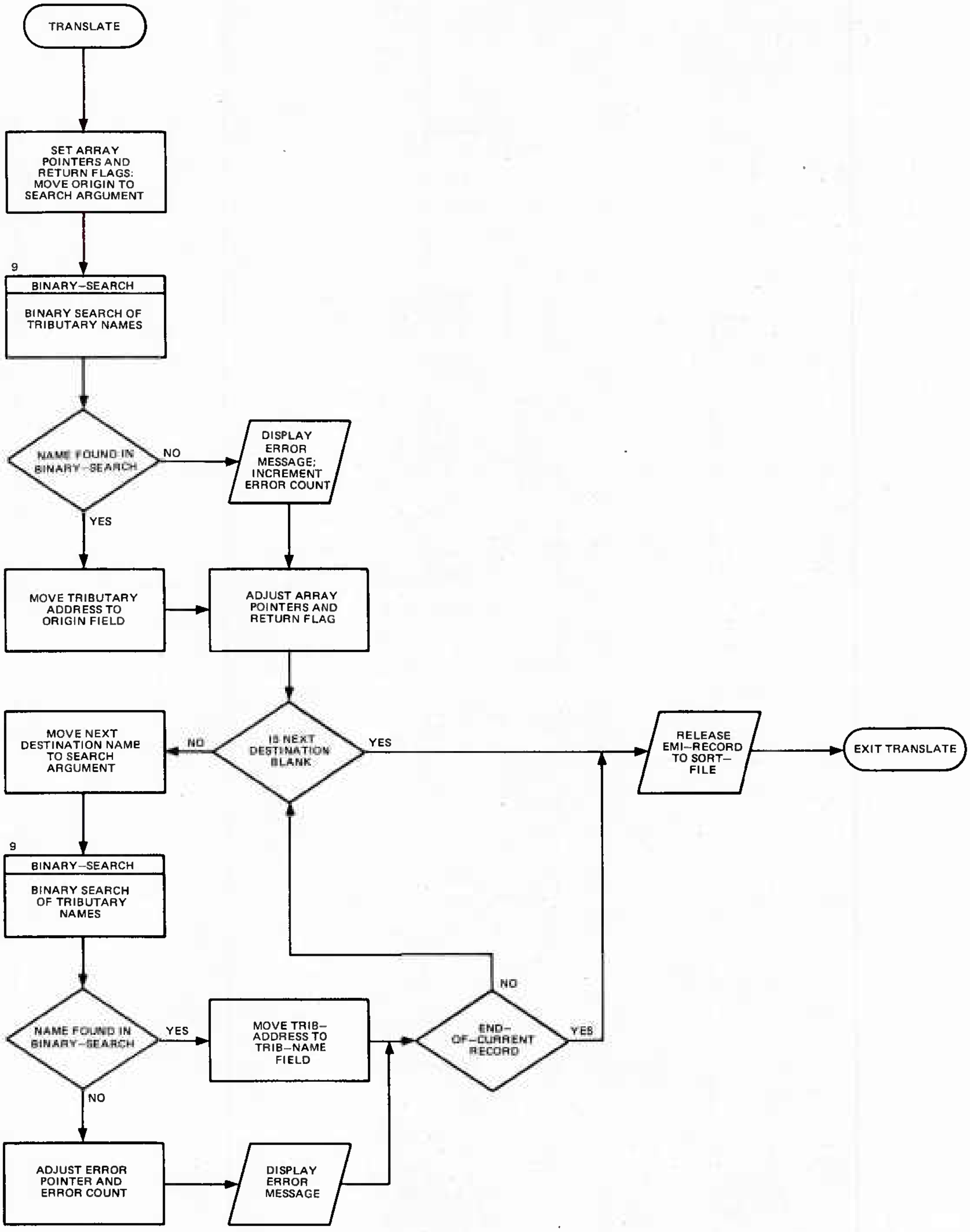
Message Sort Program (Sheet 5 of 9)


4 H
4 I
4 K
L
M
MOVE EVENT–TYPE TO EMI–RECORD: SET RESTORE FLAG
MOVE EVENT–TYPE TO EMI–RECORD: SET RESTORE FLAG
MOVE TYPE, SEQUENCE, TIME TO EMI–RECORD
NO
NODE NAME FOUND
YES
ALT–LINK FIELD BLANK
YES
7 S
NO
MOVE SEQUENCE, TIME TO EMI–RECORD: SET RETURN FLAGS
MOVE NODE NAME TO HOLD F
SET RETURN FLAGS MOVE NODE–NAME TO HOLD F
MOVE NODE ADDRESS TO TO EMI–RECORD
MOVE LINK, NODE NAMES TO HOLD 1 HOLD F: SET RETURN FLAGS
9 SEARCH–NODES
SEARCH FOR NODE NAME IN TRANSLATE TABLES
9 SEARCH–NODES
SEARCH FOR NODE–NAME IN TRANSLATE TABLES
INCREMENT ERROR COUNT: MOVE ERROR–CHARACTERS TO EMI–RECORD
9 SEARCH–LINKS
SEARCH FOR LINK. NODE NAMES IN LINK TABLE
YES
NODE NAME FOUND
NO
INCREMENT ERROR COUNT; MOVE ERROR–CHARACTERS TO EMI–RECORD
NO
NODE NAME FOUND
YES
LINK FIELD BLANK
YES
NO
NO
LINK FOUND
YES
MOVE NODE ADDRESS TO EMI–RECORD
EVENT TYPE
NODE–IN
NODE OUT
MOVE NODE ADDRESS TO EMI–RECORD
MOVE LINK NAME, NODE TO HOLD 1 HOLD F: SET RETURN FLAG
MOVE LINK ADDRESS TO EMI–RECORD
7 S
TIME INDICATED FOR RESTORAL
NO
7 S
INCREMENT ERROR–COUNT; MOVE ERROR–CHARACTERS TO EMI–RECORD
9 SEARCH–LINKS
SEARCH FOR LINK, NODE NAMES IN LINK TABLES
MOVE ERROR–CHARACTERS TO EMI–RECORD
7 S
YES
SET RETURN FLAG FOR RESTORAL
DUPLICATE NODE–IN FROM NODE–OUT
MOVE DESTINATION NODE NAME TO HOLD F
LINK FOUND
YES
MOVE LINK ADDRESS TO EMI–RECORD
NO
7 J
9 SEARCH–NODES
SEARCH FOR NODE NAME IN TRANS–LATE TABLES
MOVE ERROR–CHARACTERS TO EMI–RECORD
M
L

Message Sort Program (Sheet 6 of 9)

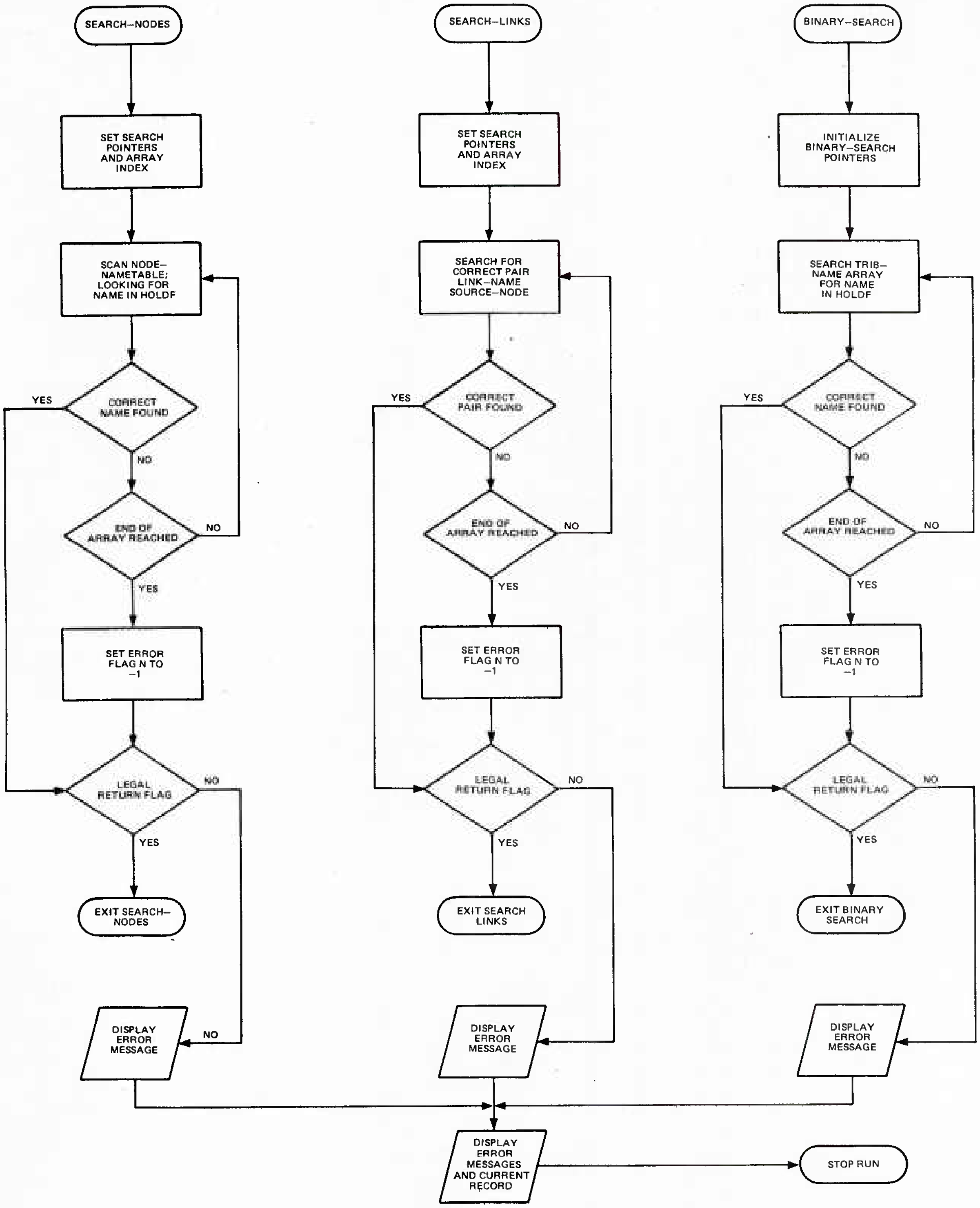

4
N
MOVE TYPE, TIME, SEQUENCE TO EMI–RECORD: SET RETURN FLAG
MOVE LINK, NODE NAMES TO HOLD 1, HOLD F
9
SEARCH–LINKS
SEARCH FOR LINK, NODE NAMES IN LINK TABLES
LINK FOUND
NO
YES
MOVE LINK ADDRESS TO EMI–RECORD
MOVE ERROR–CHARACTERS TO EMI–RECORD
SET ALTROUTE INDICATORS
S
4
O
MOVE TYPE, TIME, SEQUENCE TO EMI–RECORD: SET RETURN FLAG
MOVE LINK NAME TO HOLD F
SEARCH FOR ANY OCCURENCE OF LINK NAME IN LINK TABLES
LINK FOUND
NO
YES
MOVE LINK ADDRESS TO EMI–RECORD
S
4
P
SET ACTION–TYPE CODE TO 0
SET RETURN FLAG; MOVE ORIGIN–TRIB TO HOLD F
9
BINARY–SEARCH
SEARCH FOR TRIB–NAME IN TRIB–TABLES
9
BINARY–SEARCH
SEARCH FOR TRIB–NAME IN TRIB TABLES
TRIBUTARY FOUND
NO
YES
MOVE TRIB ADDRESS TO EMI–RECORD
MOVE ERROR–CHARACTERS TO EMI–RECORD
4
Q
SET ACTION–TYPE CODE TO 1
MOVE TIME; TYPE SEQUENCE TO EMI–RECORD
TRIBUTARY FOUND
NO
YES
MOVE TRIB ADDRESS TO EMI–RECORD
MOVE ERROR–CHARACTERS TO EMI–RECORD
SET RETURN FLAG; MOVE ALT–TRIB TO HOLD F
9
BINARY–SEARCH
SEARCH FOR TRIB–NAME IN TRIB–TABLES
TRIBUTARY FOUND
NO
YES
MOVE TRIB–ADDRESS TO EMI–RECORD
MOVE ERROR–CHARACTERS TO EMI–RECORD
R
R
SET SECURITY INDICATOR IN EMI–RECORD
6
S
RELEASE NEW EMI–RECORD TO SORT FILE
LINK–RESTORE ON
YES
5
G
NO
NODE–RESTORE ON
YES
6
J
NO
MOVE SPACES TO ALL OF EMI–RECORD
4
A

Message Sort Program (Sheet 7 of 9)



TRANSLATE
SET ARRAY POINTERS AND RETURN FLAGS: MOVE ORIGIN TO SEARCH ARGUMENT
9
BINARY–SEARCH
BINARY SEARCH OF TRIBUTARY NAMES
NAME FOUND IN BINARY–SEARCH
NO
DISPLAY ERROR MESSAGE; INCREMENT ERROR COUNT
YES
MOVE TRIBUTARY ADDRESS TO ORIGIN FIELD
ADJUST ARRAY POINTERS AND RETURN FLAG
MOVE NEXT DESTINATION NAME TO SEARCH ARGUMENT
NO
IS NEXT DESTINATION BLANK
YES
RELEASE EMI–RECORD TO SORT–FILE
EXIT TRANSLATE
9
BINARY–SEARCH
BINARY SEARCH OF TRIBUTARY NAMES
NO
NAME FOUND IN BINARY–SEARCH
YES
MOVE TRIB–ADDRESS TO TRIB–NAME FIELD
END–OF–CURRENT RECORD
YES
NO
ADJUST ERROR POINTER AND ERROR COUNT
DISPLAY ERROR MESSAGE

Message Sort Program (Sheet 8 of 9)



Message Sort Program (Sheet 9 of 9)

SECTION 6 - BASIC SIMULATION

6.1 PROGRAM REQUIREMENT DESCRIPTION

6.1.1 Program Name: Basic Simulation

6.1.2 Program Identification: DNBASIM1

6.1.3 Purpose

The Basic Simulation program performs an event-by-event simulation of a store-and-forward message system utilizing the network established in the Status Generator program and the events and messages generated by the Message Generator programs and Message Sort program. The program generates event and status descriptor records from which the Reports programs establish network performance.

6.1.4 Requirements

The program is written in FORTRAN IV language and requires a minimum of 320K bytes of memory for execution on on an IBM 360 System.

6.2 INPUT SPECIFICATIONS

Input to the Basic Simulation program is composed of three data files generated by preceding programs and a card set containing simulation parameters and options.

6.2.1 Status Table File

The Status Table file is a disk file containing a binary image of two arrays which reflect the network to be simulated. NODRAY contains the Node Status Tables developed by the Status Generator program from its input data. STORAY contains the Link Status and Tributary Status Tables also developed by the Status Generator program. Generation of this file is described in Section 2.

6.2.2 Translation File

The Translation file contains a cross reference between the Switch, Link, and Tributary names and the address identification of these items in their respective

Status Tables. The file is used in printing unusual occurrences developed during network simulation. Generation of this file is described in Section 2.

6.2.3 Event Message File

The Event Message file contains the Special Events and messages to be used in the network simulation. This input is sorted into chronological order by the Message Sort program. Generation of this file is described in Section 5.

6.2.4 Parameter and Option Cards

Eleven different parameter and option cards may be introduced into the Basic Simulation. The format of these cards is given in Tables 6-1 through 6-4.

6.2.4.1 Interval Card

The INTERVAL card is the only parameter card required by the program. It establishes the start and stop times of the simulation period.

6.2.4.2 Status Delay Cards

The STATUS DELAY SWT, STATUS DELAY LINK, and STATUS DELAY TRIB cards establish the intervals at which Switch, Link, and Tributary Status records are written on the output Status file. If any of these cards are omitted, its associated Status records are not are not written on the output file.

6.2.4.3 Checkpoint Card

The Basic Simulation program has provision for writing the program's essential core elements on an output file to restart the program at a later time. The CHECKPOINT card establishes the interval of writing these core elements.

6.2.4.4 Restart Card

The RESTART card is used when the simulator is being restarted as provided for by the CHECKPOINT card. This is the only card permitted in the restart mode. An optional entry in this card permits specification of a new stop time which replaces the time specified in the INTERVAL card.

6.2.4.5 Option Cards

Six options are provided to establish special conditions. These options are further discussed in Paragraph 6.4.

6.3 OUTPUT SPECIFICATIONS

The Basic Simulation program generates up to five output files and one listing.

6.3.1 Speed of Service (SOS) Event File

The program notes major events by writing an event descriptor on an appropriate file. The descriptor record contains an action code to specify the type of event plus other information such as message number, message length, message precedence, message format, switch address, link or tributary address, and link or channel number. To conserve file space, the time of an occurrence is not included in each descriptor. Instead, a time record with a specific action code is written on these files as the simulator clock is updated. The time record cannot occur more than once a second. The format of the event record is given in Table 6-5.

These records are used by other programs in producing the reports of network performance. The SOS Event file contains descriptors for Action Code 1 (message entering the system) and Action Code 6 (message delivery to its destination) and is used for producing SOS Reports.

6.3.2 Link Event File

The Link Event file is similar to the SOS Event file and contains descriptors for Action Code 2 (message arrival at a Switch from a tributary), Action Code 3 (message transmission initiated on a link), Action Code 4 (message arrival from a link), Action Code 5 (message transmission initiated to a tributary), Action Code 7 (message preemption on a link), Action Code 8 (link channel outage), Action Code 9 (link channel restoral) and Action Code A (message placed in tributary queue). This file is used for producing Link Utilization, In-Station Handling Time, and Network Delay Reports.

6.3.3 Status File

The program is designed to periodically provide status reports on the network. Essentially, these reports are a summary of the queues of messages awaiting transmission on tributaries, links, and total messages in queue at a switch; and are written as Status Records on the Status file. The format for these records is given in Table 6-6. This file is used for producing the Switch, Link, and Tributary Queue Reports.

6.3.4 Dump Files

To provide restart capabilities, the essential core elements of the program may be written periodically. To assure having at least one restart file available, two files are written, not simultaneously, but alternately. At the first checkpoint Dump file 1 is written, at the second checkpoint Dump file 2 is written, at the third Dump file 1 is overwritten, at the fourth Dump file 2 is overwritten, et cetera. Upon successful completion of the desired simulation period, the files are no longer required and are discarded. The files are in binary image and therefore have no format.

6.3.5 Printout

The printout from the Basic Simulation program is in three parts. The first reproduces the parameter and option cards and any associated error diagnostics. When the checkpoint parameter is specified, the second part reflects the time a checkpoint was taken and on which Dump file the core elements were written. During simulation, a file of unusual occurrences is maintained. At the end of simulation, this file is printed as part three. This listing reflects such occurences as Special Events and their results, LTC throttling and unthrottling, and messages dropped due to low security on a link or tributary.

6.4 PROGRAM LOGIC

6.4.1 Tables

Each element in a network being simulated is represented by a table. These tables contain information such as pointers to other related elements, queue lists, queue counters, parameters and characteristics of an element, pointers to events related to the element, and indicator flags reflecting current status of the element. There are three types of permanent tables which are constructed by the Status Generator program from input parameters and passed to the Basic Simulation program. These are Node Status, Link Status, and Tributary Status Tables. Channel Status Tables are included as subdivisions of the Link Status and Tributary Status Tables. There are four types of temporary tables generated by the program as the need arises and which are deleted when their function has been completed. These are Message Status, Message Continuation, Event Status, and Queue Linkage Tables. These tables are described in Tables 6-7 through 6-13. To make the program more computer independent and to facilitate program expansion, all program references to table items are made via mnemonics and pseudo-instructions. The program instructions are then processed by a special FORTRAN Preprocessor program in which the actual table structure has been defined. The preprocessor replaces the pseudo-instructions with valid instructions for the particular computer on which the model is being run. Section 15 describes the preprocessor and Appendix D defines the current table structures.

6.4.2 Calendar of Events

Except for the initialization and wrapup functions, the Basic Simulation program operation does not follow any logical sequence of steps. Its actions are triggered by the events introduced via the Event Message file. These events, in turn, generate other events which are to be executed at a later time as determined by the characteristics of the initial event and/or the element of the network to which it applies. Appendix B contains a discussion of the various event types. Appendix A contains a synopsis of the events which would take place to transmit a message through a network.

Program operation is controlled by a calendar of events which is a queue of waiting events arranged in time sequence. Program operation is asynchronous, i. e., when the program finishes one task, it retrieves the next event from the calendar, updates the simulator clock, and performs the desired function. This simulator function is normally equivalent to the end of some real network operation which takes a certain amount of time to perform. Thus, in simulating a function, the start of an operation normally generates an end of operation event which is to occur later and is therefore entered into the calendar. The start of an operation is generally triggered by an event waiting in queue.

6.4.3 Initialization Subroutine

When the Basic Simulation program is executed, the first subroutine called is INIT. Since this routine is only used once, it is later overlaid by another subroutine to conserve core space. The subroutine first reads the Status Tables generated by the Status Generator program, then reads the parameter and option cards. The calendar of events is initialized and an end of simulation event based on the stop time in the INTERVAL card is generated and placed in the calendar. When the Status Records and restart capabilities are desired, appropriate control events are generated and placed in the calendar. When operating in the normal cyclic mode, a Message Processor Cycle event is generated for each switch. Each Link Status Table and Tributary Status Table is then searched to see if a preload queue delay was specified. When so specified, a preload release event is generated and placed in the calendar.

If the program is being restarted from a previous run, the subroutine reads in the core elements from the Restart file, repositions input and output files to the correct point, and, if a new stop time was specified, cancels the previous end of simulation event and generates a new one.

6.4.4 Message Processor Cycle

The cyclic functions of an AUTODIN switch are simulated by Message Processor (MP) subroutines in the program. These subroutines are Process Input Data (PID), Process Output Data (POD), Release Output Data (RELEAS), and Miscellaneous functions. Execution of these routines is controlled by an event entered in the calendar of events. During initialization one of these events is created for each switch and the MP program pointer is set to PID. When the event is executed a test is made to see if there is any input data to process. When none is present, the overhead delay time is added to the current time in the event, the program pointer is advanced to indicate the next MP program, and the event is returned to the calendar. When there is data to be processed, Subroutine PID is called. When processing is completed, the PID overhead delay time plus any additional delay time caused by message processing is added to the current time in the event, the pointer is updated, and the event is returned to the calendar. Similar steps are taken for POD and RELEAS Subroutines. The model does not perform any tasks during the Miscellaneous functions. Its purpose is to account for the time taken in a switch cycle for any functions not simulated. When executed, the miscellaneous overhead delay is added, the program pointer is reset to POD, and the event returned to the calendar.

6.4.4.1 Process Input Data (PID) Subroutine

The PID Subroutine can perform three tasks and is only entered if there is a task to be performed. The three tasks are removing messages from the tape queue list if the switch has processing space, accepting incoming messages from tributaries and other switches, and resuming tributary transmissions which were blocked by overloaded LTC/ADUs. To improve program running time, a change concepts from the previous model was used. Rather than search all of a switch's input channels for messages which are ready to enter when a transmission event has been completed, that event is entered in an input action list. The program removes these events from the action list and when processing space is available, generates a routing event for the message. If processing space is not available, the message is added to the tape queue.

6.4.4.2 Process Output Data (POD) Subroutine

This subroutine schedules the outgoing transmission of messages. Whenever a message is in queue, it checks the appropriate output channels for one which is available or, in the case of high precedence messages, for one which may be preempted. If transmission can be scheduled, an appropriate transmission event is generated and placed in a release list. Again, to improve program running time rather than search all outgoing links and tributaries for messages in queue, an output action list is maintained. Whenever a message is placed in queue, that link or tributary is placed in an output action list if not already there.

6.4.5 Option Cards

The option cards are used either to invoke certain options or to disable some feature which is standard in the AUTODIN network but which may not apply to other store-and-forward networks.

6.4.5.1 EDIT Option Card

The EDIT option card disables the normal cyclic functions at a switch, and initiates the Event Determined Interval Timing (EDIT) mode. Appendix A provides a discussion of this mode of operation.

6.4.5.2 NO ETR Option Card

The EDIT mode utilizes an Effective Transmission Rate (ETR) factor in determining the time required to transmit a message. Under normal operation the time required is a straight computation of rate time length. In the EDIT mode, the rate is a nonlinear function of the message length. This feature may be disabled using the NO ETR option card.

6.4.5.3 NO DUPS Option Card

When the Statistical Message Generator is used to provide message input, it is possible that a given tributary may occur more than once as the destination in multiple addressed messages. Under these circumstances the message will be delivered more than one time to the same tributary. If this is not desirable, the NO DUPS option card eliminates duplicate deliveries.

6.4.5.4 NO ALT Option Card

It may be desirable to inhibit altrouting of messages during a link outage for some particular simulation problem. This can be inhibited with a NO ALT option card.

6.4.5.5 NO MCB Option Card

In normal message processing in the AUTODIN network, a Message Control Block (MCB) is added to each message when it is sent across a link. It may be desirable to disable this feature if a non-AUTODIN network is being simulated. A No MCB option card cancels this operation.

6.4.5.6 NO PRE Option Card

For the simulator to rapidly assume a fully loaded condition, a preload delay time may be optionally applied to each of the network tributaries and links. When this option is exercised, any low priority messages placed in the output queues are not released until this delay expires.

When the Tributary and Link files are assembled by the Traffic Preprocessing Section, an average queue time is computed and applied to the network. To cancel this delay a NO PRE option card may be used.

6.4.6 Memory Layout

The main segment and most of the subroutines which comprise the Basic Simulation program are resident in core at all times. The Initialization and Wrapup Subroutines are not required full time; therefore, the Initialization Subroutine is later overlaid by the Special Events Subroutine which in turn is overlaid at the end of the run by the Wrapup Subroutine. There are two different versions of the Process Output Data Subroutine, one used in the normal mode and the other in the EDIT mode. These two subroutines are also arranged in an overlay segment to conserve memory. The memory layout is depicted in Figure 6-1.

Program logic flow charts are included at the end of this section.

6.5 MACHINE DEPENDENCE

Since the program is written in FORTRAN IV language, it is essentially machine independent. All references which depend upon a particular bit configuration of a computer word are made with mnemonic references and pseudo-instructions.

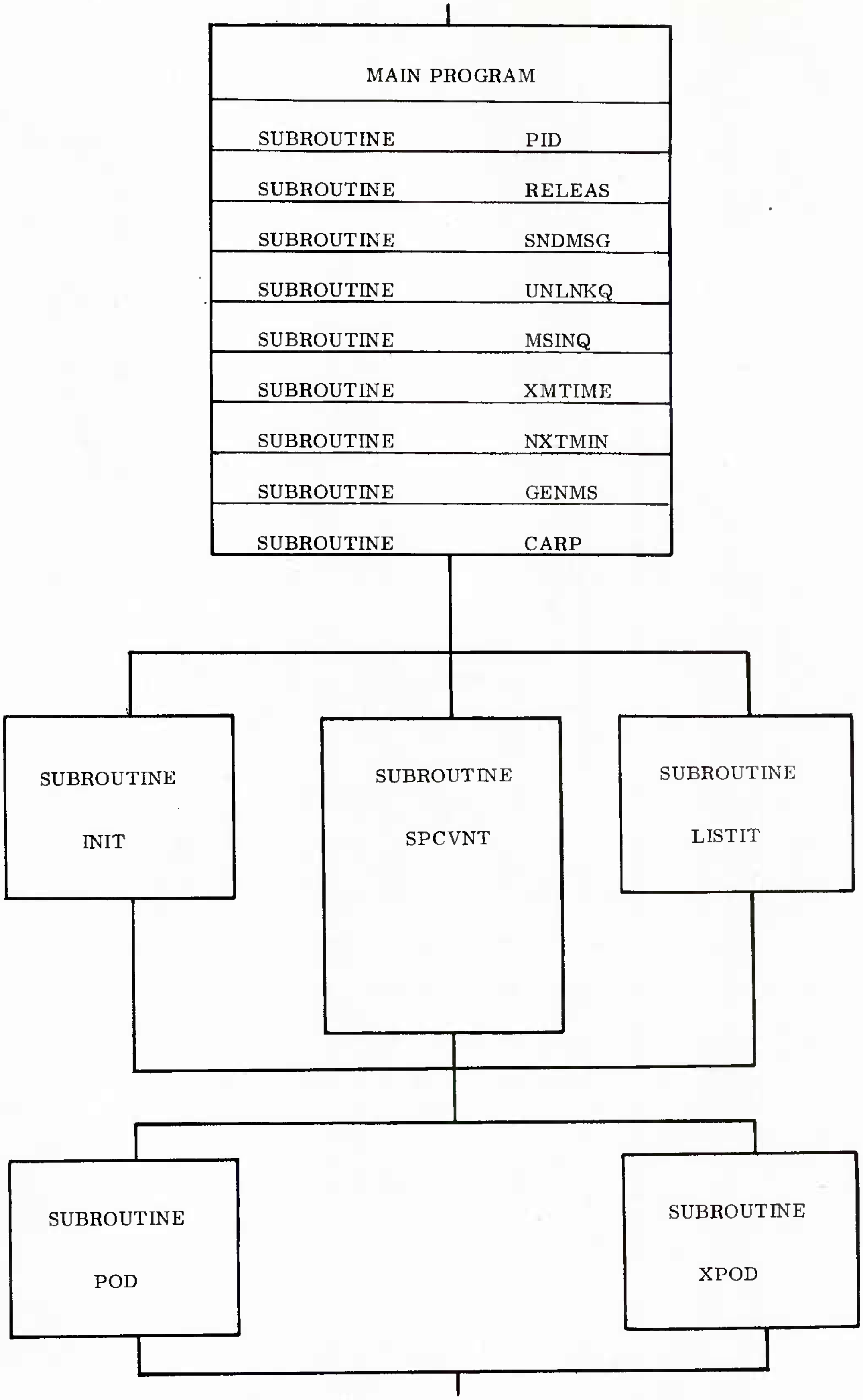


Figure 6-1. Basic Simulation Memory Layout

Table 6-1. Basic Simulation Interval Card Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-8	Type of Card	INTERVAL
9-20	Blank	
21-24	Start Time (HHMM)	Start of simulation period in military time.
25	Blank	
26-29	Stop Time (HHMM)	Stop simulation period in military time.
30-80	Blank	

Table 6-2. Basic Simulation Parameter Card Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-12	Type of Card	STATUS DELAY or CHECKPOINT
13	Blank	
14-17	Type of Delay	If card type is STATUS DELAY contains SWT, LINK, or TRIB
18-20	Blank	
21-24	Time Interval in Minutes	Specifies interval between Status records or writing of restart files. (Field is right adjusted, leading zeroes not required).
25-80	Blank	

Table 6-3. Basic Simulation Option Card Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-7	Type of Card	EDIT, NO ETR, NO DUPS, NO ALT, NO MCB or NO PRE
8-80	Blank	

Table 6-4. Basic Simulation Restarts Card Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-7	Type of Card	RESTART
8-20	Blank	
21-24	Stop Time (HHMM)	New Stop time expressed in military time (optional).
25-80	Blank	

Table 6-5. Event Record Format

BYTE POSITION	MNEMONIC REFERENCE OR ITEM NAME	USAGE
1	ORAC	Action code which identifies event type: 0 = Time record 1 = Message entering network 2 = End of transmission from tributary to a switch 3 = Start of message transmission on a link 4 = End of message transmission on a link 5 = Start of transmission from switch to tributary 6 = Message arrival at destination 7 = Message preemption on a link 8 = Link channel outage 9 = Link channel restoral A = Message placed in tributary queue
2-7	ORT	Time in seconds for a time record (AC=0)
	or	
	ORNUM	Message number (AC=1, 2, 3, 4, 5, 6, 7 or A)
8-13	ORTRIB	Tributary Identification (AC=1, 2, 5, 6, or A)
	or	
	ORLINK	Link Identification (AC=3, 4, 7, 8 or 9)
14-16	ORLN	Message Length (AC=1, 2, 3, 4, 5, 6, 7 or A)
17-19	ORDES	Number of Destination (AC=1)
	or	
	ORCH	Channel Number (AC=2, 3, 4, 5, 6, 7, 8, 9)
20	ORPC	Message Precedence (AC=1, 2, 3, 4, 5, 6, 7 or A)
21-22	ORSW	Switch Number (AC=1, 2, 3, 4, 5, 6, 7, 8, 9 or A)
23	ORTP	Message Type (AC=1, 2, 3, 4, 5, 6, 7 or A)

Table 6-6. Status Record Format

BYTE POSITION	MNEMONIC REFERENCE OR ITEM NAME	USAGE
1	ORACS	Action code which identifies record type: 0 = time 1 = Switch Status 2 = Link Status 3 = Trib Status 4 = LTC Status 5 = Fixed Queue Status
2-7	ORTS	Time in seconds (AC=0)
	or	
	ORQL	Number of link messages in queue (AC=1)
	or	
	ORLNK	Link identification (AC=2)
	or	
	ORTRB	Tributary identification (AC=3)
	or	
	ORLTC	LTC Number (AC=4)
	or	
	ORFQS	Queue entries available (AC=5)
8-13	ORQIS	Number of trib input messages in queue (AC=1, 3)
	or	
	ORQHI	Number of high precedence messages in queue (AC=2)
	or	
	ORLBA	Lineblocks Available (AC=4)
	or	
	ORTQC	Messages in Tape Queue (AC=5)
14-19	ORQOS	Number of trib output messages in queue (AC=1, 3)
	or	
	ORQLOS	Number of low precedence messages in queue (AC=2)
	or	
	ORTIS	Throttle Indicator (AC=4)
20-21	ORNDS	Switch Number (AC-1, 2, 3, 4, 5)

Table 6-7. Node Status Table

MNEMONIC REFERENCE	ITEM NAME	ITEM USAGE
NSNO	Switch Outage Indicator	Set to ∅ if switch is in service, set to 1 if out of service
NSNL	Number of Links	Number of links connected to switch*
NSNL	First Link	Address of the first link*
NSNT	Number of Tribs	Number of tributaries connected to switch*
NSFT	First Trib	Address of the first link*
NSNLT	Number of LTCs	Number of LTC/ADUs assigned to switch*
NSRT	Routing Table	Address of first entry in switches· Routing Table
NSQIT	Input Queue	Number of messages queued to tributary input channels
NSQOT	Output Queue	Number of messages queued to tributary output channels
NSQL	Link Queue	Number of messages queued to link channels
MPLTCD	LTC Overhead Delay	Time in milliseconds to initiate an LTC/ADU data transfer*
MPLTCX	LTC Transfer Delay	Time in microseconds to transfer 1 line block to or from an LTC/ADU*
MPIPID	PID Overhead Delay	Time in milliseconds for each PID section of MP cycle*
MPISOM	PID Message Delay	Additional time in milliseconds added for each message processed in PID*
MPOPOD	POD Overhead Delay	Time in milliseconds for each POD section of MP cycle*
MPOSOM	POD Message Delay	Additional time in milliseconds added for each message processed in POD*
MPOMX	Message Exchange Delay	Time in microseconds per line block to convert message format. Total is added to POD Delay*
MPMISC	Miscellaneous Delay	Time in milliseconds for other processing in MP cycle*
MPIRIH	Routing Delay High Priority	Time in milliseconds required to perform high priority message routing*
MPIRIL	Routing Delay Low Priority	Time in milliseconds required to perform low priority message routing*
MPOFQ	Fixed Queue Limit	Limit of messages which may be queued to a switches output channel**

Table 6-7. Node Status Table (Continued)

MNEMONIC REFERENCE	ITEM NAME	ITEM USAGE
MPNFQ	No Queue Indicator	Set to 1 if no fixed queue limit at a switch*
MPNLTC	No LTC Indicator	Set to 1 if LTC/ADU functions are not simulated*
MPICS	Input Action List Start	Address of first event in PID action list
MPICE	Input Action List End	Address of last event in PID action list
MPOQS	Output Action List Start	Address of first event in POD action list
MPOQE	Output Action List End	Address of last event in POD action list
MPIQS	Input Trib List Start	Address of first trib event in PID list
MPIQE	Input Trib List End	Address of last trib event in PID list
MPTQS	Tape Queue Start	Address of first message in Tape Queue List
MPTQE	Tape Queue End	Address of last message in Tape Queue List
MPTQC	Tape Queue Counter	Messages in Tape Queue
MPORL	Output Release List	Address of first event in Output Channel Release List
MPCVNT	MP Event	Address of MP Cycle Event
MPPROG	Program Pointer	Points to next program in MP Cycle 1 POD, 2 = RELEASE, 3 = PID, 4 = MISC
FETT	LTC Throttle Threshold	An LTC/ADU becomes throttled if its line blocks available falls below this value*
FETU	LTC Unthrottle Threshold	An LTC/ADU resumes operation when its line blocks available rises above this value*
FELBA1	Line Blocks Available	Line blocks available at LTC/ADU No. 1 **
FETI1	Throttle Indicator	Throttle indicator for LTC/ADU No. 1
FELBA2	Line Blocks Available	Line blocks available at LTC/ADU No. 2 **
FETI2	Throttle Indicator	Throttle indicator for LTC/ADU No. 2
FELBA3	Line Blocks Available	Line blocks available at LTC/ADU No. 3**
FETI3	Throttle Indicator	Throttle indicator for LTC/ADU 3
FELBA4	Line Blocks Available	Line blocks available at LTC/ADU No. 4
FETI4	Throttle Indicator	Throttle Indicator for LTC/ADU No. 4

* Items set by Status Generator Program

** Items set initially by Status Generator Program but later altered by Basic Simulation

Table 6-8. Link/Tributary Status Table

MNEMONIC REFERENCE	ITEM NAME	ITEM USAGE
LTLO	Link Outage Indicator	Set to ∅ if in service, set to 1 if all channels out of service
LTNC	Number of Channels	Number of channels assigned*
LTLTC	LTC Number	Indicates to which LTC/ADU the link is connected. ∅ = LTC No. 1, 1 = LTC No. 2, etc*
LTSN	Source Switch	Number of switch to which connected*
LTTN	Tributary Flag	Status Table refers to a tributary, set to 1*
LTSC	Security	Highest security level permitted Special = 4 Top Secret = 3 Secret = 2 Confidential = 1 Unclassified = ∅
LTCA	Channels Available	Number of output channels available **
LTCB	Channels Busy	Number of output channels busy
LTPRE	Preload Flag	Set to 1 if there is a preload queue delay for low precedence messages. Set to ∅ when delay expires. **
LTQS	Queue Start	Address of first message in output queue. (When preload delay is specified, contains number of seconds of delay desired*)
LTQE	Queue End	Address of the last message in output queue
LTQP	Queue Priority	Address of the last priority message in output queue
LTOIP	Preempt Flag	When set to 1, Operational Immediate messages are classed as high priority and have preempt capability (set only by a special event)
LTQF	Queue Flag	When entered in MP Output Queue List, set to 1
	ITEMS IN LINK STATUS ONLY	
LNDN	Destination Node	Number of destination node*
LNDL	Destination Link	Address of Status Table of reverse direction of this Link*
LNQHI	High Queue	Count of messages of high priority queued to output channels
LNQLO	Low Queue	Count of messages of low priority queued to output channels

Table 6-8. Link/Tributary Status Table (Continued)

MNEMONIC REFERENCE	ITEM NAME	ITEM USAGE
LNAA	Altroute Indicator	When set to 1, indicates that messages of all priorities may be altrouted. (Set only by a special event.)
	ITEMS IN TRIBUTARY STATUS ONLY	
TROQ	Output Queue Counter	Count of messages queued to output channels
TRIQ	Input Queue Counter	Count of messages queued to input channels
TRINQ	Input Queue List	Address of first message in queue to input channels
TRVN	VON Dial Time	If tributary is connected to switch via an AUTOVON line, contains time required to dial up the line*
TRSPT	Teletype Speed	Transmission speed for teletype messages*
TRSPC	Card Speed	Transmission speed for punched card messages*
TRSPM	Mag Tape Speed	Transmission speed for magentic tape or data pattern messages*
TRAD	Acknowledgement Delay	Acknowledgement delay time to receipt for message*
TRQFI	Input Queue Flag	Set to 1, it has been entered in MP Input Queue List

* Items set by Status Generator Program

** Items set initially by Status Generator Program but later altered by Basic Simulation

Table 6-9. Channel Status Table

MNEMONIC REFERENCE	ITEM NAME	ITEM USAGE
CHEV	Event Address	When channel is in use, contains event address
CHCO	Channel Outage Indicator	Set to Ø if channel is in service, set to 1 if not
CHSM	SOM Flag	Set to 1 when a message is assigned to a channel Not cleared until channel is released
CHEM	EOM Flag	Set to 1 when message transmission is complete
CHPR	Preempt Flag	Set to 1 when a preempt has been requested on a channel
CHPRA	Preempt Acknowledgement Flag	Set to 1 when a preempt request has been acknowledged
CHDFLG	Acknowledgement Delay Flag	Set when either specified acknowedgement delay has expired or when message being transmitted has been accepted by destination, whichever occurs first. The channel is released when the action occurs.
	ITEMS IN LINK CHANNEL STATUS ONLY	
LCSP	Channel Speed	Transmission Speed of link channel*
LCAD	Acknowledgement Delay	Acknowledgement delay time to receipt for message*

* Items set by Status Generator Program

Table 6-10. Message Status Table

MNEMONIC REFERENCE	ITEM NAME	ITEM USAGE
MSNUM	Message Number	Number which should be a unique message identification
MSRI	Number of Destinations	Number of destination tributary address for message
MSLN	Message Length	Length in line blocks
MSTP	Message Type	Message type: Teletype = 1, Card = 2, Data = 3
MSPR	Message Priority	Message priority: Flash Override = 1, Flash = 2, Operational Immediate = 3, Priority = 4, Routine = 5.
MSSC	Message Security	Message security: Special = 4, Top Secret = 3, Secret = 2, Confidential = 1, Unclassified = 0
MSXF	Transmission Flag	Set to 1 when message is assigned to a channel for transmission
MSLR	Linkage Field	When messages are linked together in a queue or list, contains address of next message in chain
MSDN1	Destination One	Address of 1st destination tributary
MSDN2	Destination Two	If MSRI = 1 - not used; if MSRI = 2 - contains 2nd destination; if MSRI > 3 - contains address of Message Continuation Table

Table 6-11. Message Continuation Table

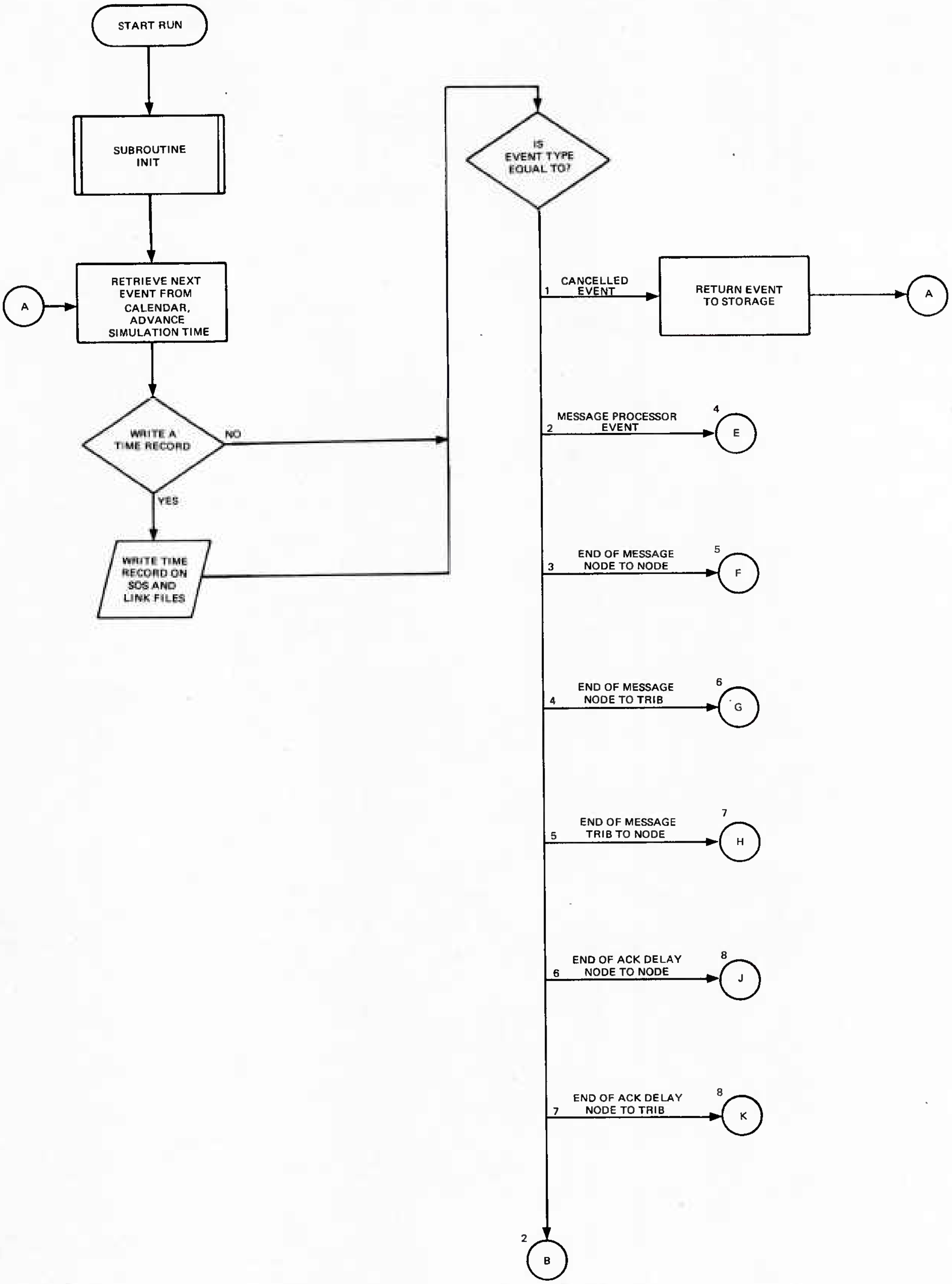
MNEMONIC REFERENCE	ITEM NAME	ITEM USAGE
MCDN1	Destination 2	Address of destination tributary (2nd, 7th, etc.)
MCDN2	Destination 3	Address of destination tributary (3rd, 8th, etc.)
MCDN3	Destination 4	Address of destination tributary (4th, 9th, etc.)
MCDN4	Destination 5	Address of destination tributary (5th, 10th, etc.)
MCDN5	Destination 6	Address of destination tributary (6th, 11th, etc.)
MCDN6	Destination 7	Address of last destination tributary or next message continuation Table

Table 6-12. Transmission Event Status Table

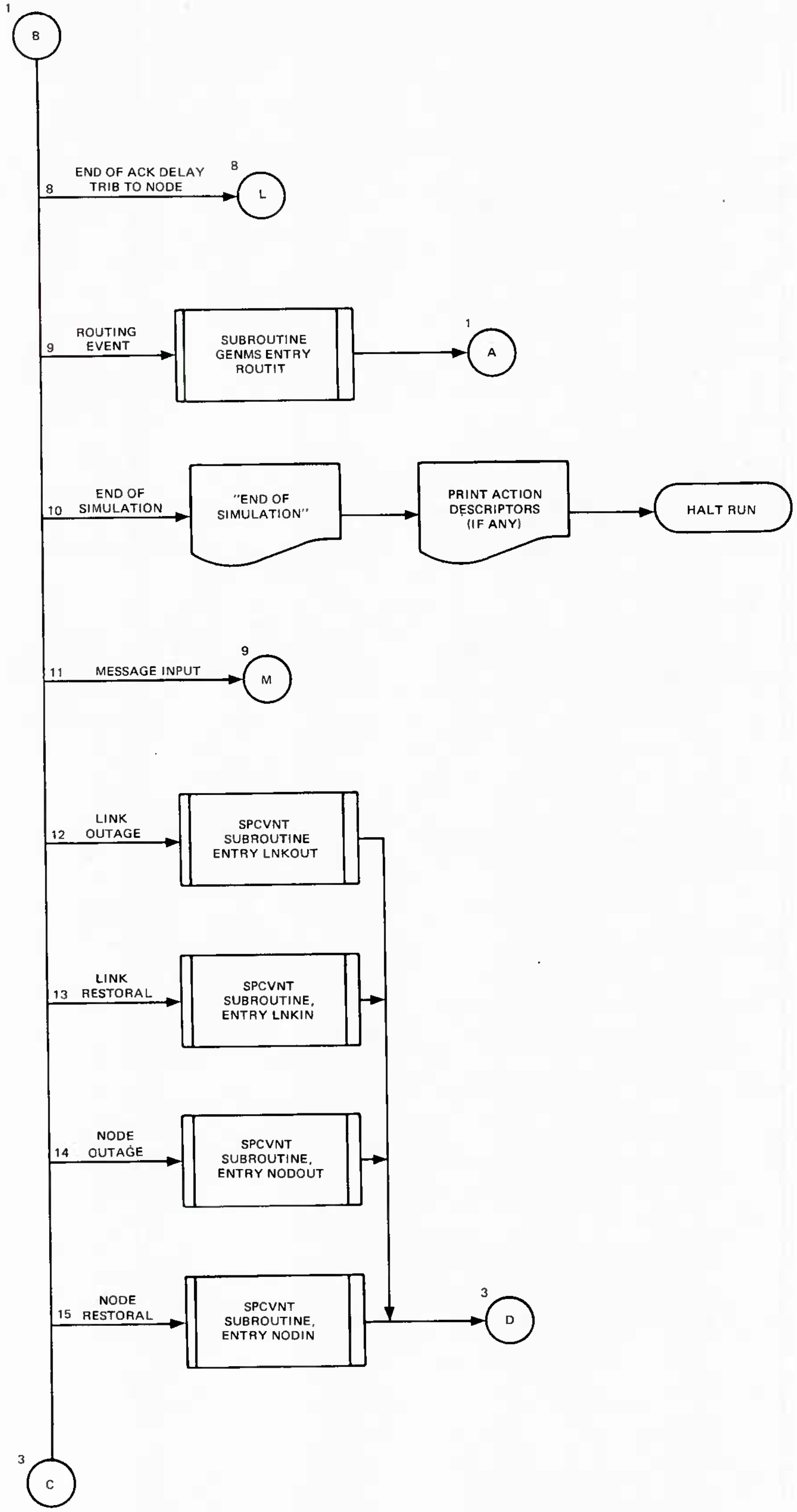
MNEMONIC REFERENCE	ITEM NAME	ITEM USAGE
EVBL	Linkage	When events are linked together in a queue or list, contains address of next event in chain
EVNS	Switch Code	Switch number to which event applies
EVT	Event Time	Time event is to take place. Only used if event is placed in calendar of events.
EVMS	Message Pointer	When an event applies to a message transmission, contains Message Status Table address
EVLSA	Link Pointer	When an event applies to an action on a link or tributary contains Link or Tributary Status Table address.
EVCS	Channel Pointer	When an event applies to an action on a link or tributary channel, contains channel number.
EVTP	Event Type	Indicates event type

Table 6-13. Queue Linkage Table

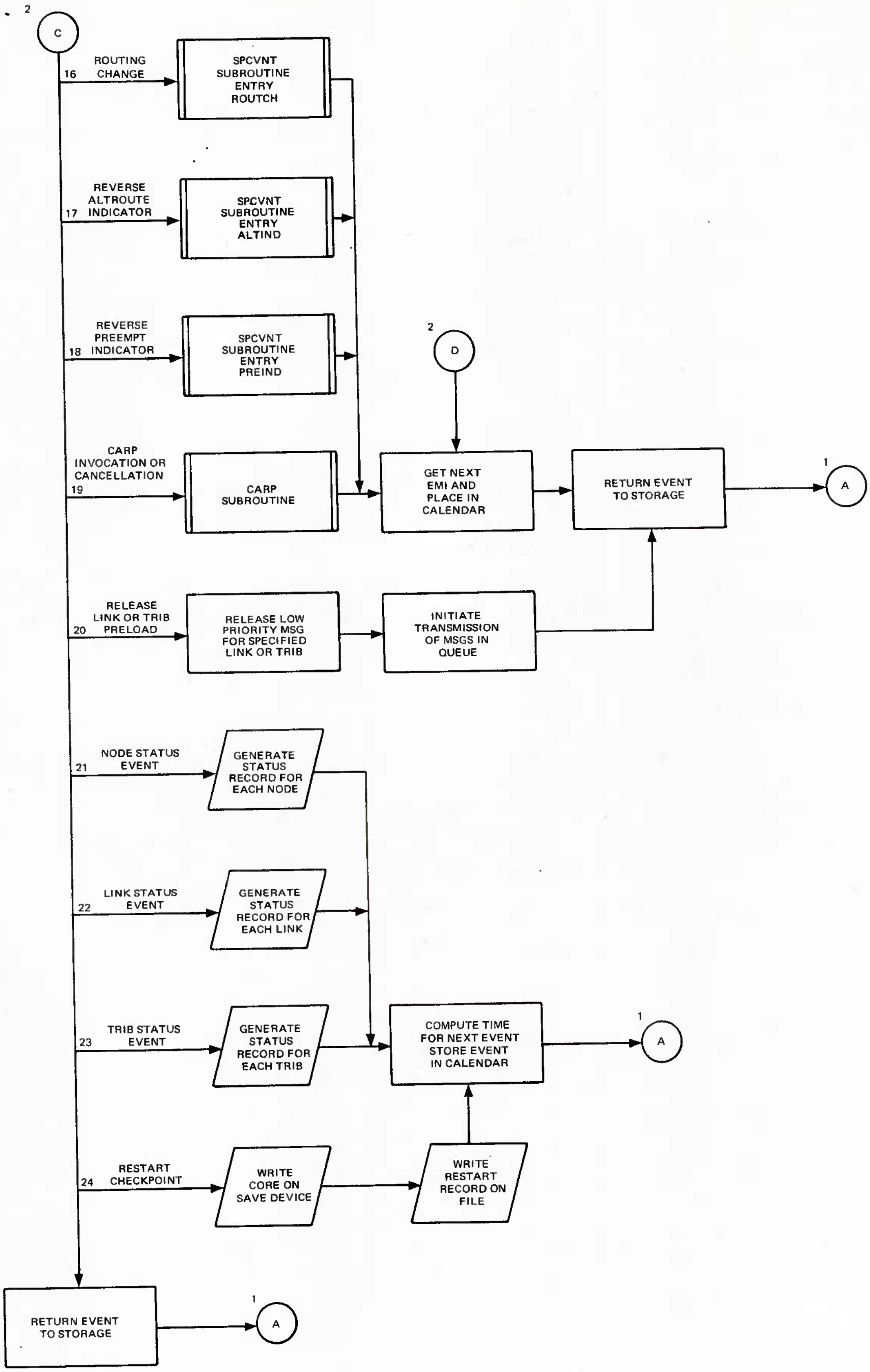
MNEMONIC REFERENCE	ITEM NAME	ITEM USAGE
EVBL	Linkage	Address of next item in queue
EVLK	Pointer Address	Address of Status Table entered in queue


START RUN
SUBROUTINE INIT
A
RETRIEVE NEXT EVENT FROM CALENDAR, ADVANCE SIMULATION TIME
WRITE A TIME RECORD
NO
YES
WRITE TIME RECORD ON SOS AND LINK FILES
IS EVENT TYPE EQUAL TO?
1 CANCELLED EVENT
RETURN EVENT TO STORAGE
A
2 MESSAGE PROCESSOR EVENT
4 E
3 END OF MESSAGE NODE TO NODE
5 F
4 END OF MESSAGE NODE TO TRIB
6 G
5 END OF MESSAGE TRIB TO NODE
7 H
6 END OF ACK DELAY NODE TO NODE
8 J
7 END OF ACK DELAY NODE TO TRIB
8 K
2 B

Basic Simulation Main Program (Sheet 1 of 9)

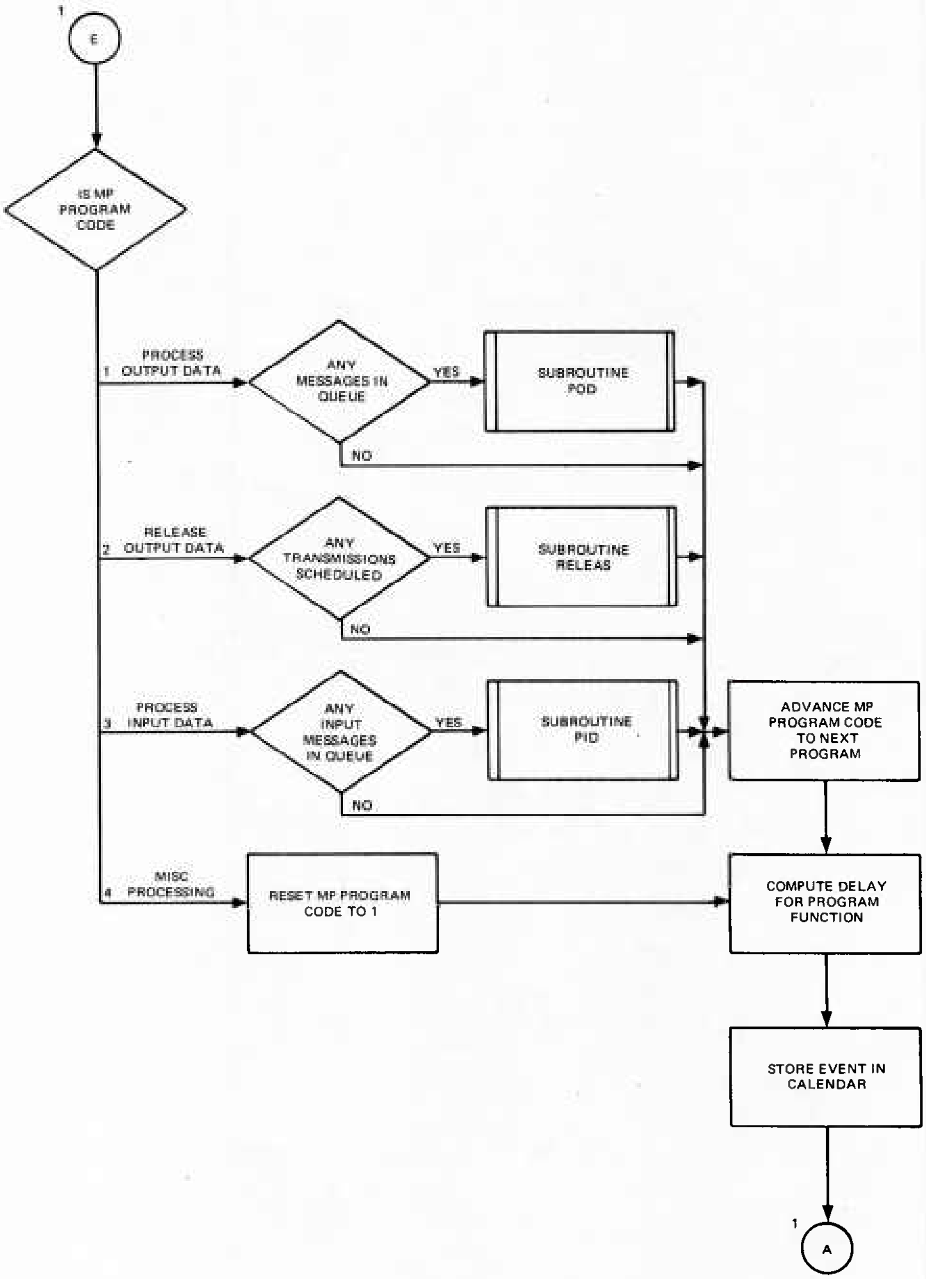


Basic Simulation Main Program (Sheet 2 of 9)


2
C
16 ROUTING CHANGE
SPCVNT SUBROUTINE ENTRY ROUTCH
17 REVERSE ALTROUTE INDICATOR
SPCVNT SUBROUTINE ENTRY ALTIND
18 REVERSE PREEMPT INDICATOR
SPCVNT SUBROUTINE ENTRY PREIND
2
D
19 CARP INVOCATION OR CANCELLATION
CARP SUBROUTINE
GET NEXT EMI AND PLACE IN CALENDAR
RETURN EVENT TO STORAGE
1
A
20 RELEASE LINK OR TRIB PRELOAD
RELEASE LOW PRIORITY MSG FOR SPECIFIED LINK OR TRIB
INITIATE TRANSMISSION OF MSGS IN QUEUE
21 NODE STATUS EVENT
GENERATE STATUS RECORD FOR EACH NODE
22 LINK STATUS EVENT
GENERATE STATUS RECORD FOR EACH LINK
23 TRIB STATUS EVENT
GENERATE STATUS RECORD FOR EACH TRIB
COMPUTE TIME FOR NEXT EVENT STORE EVENT IN CALENDAR
1
A
24 RESTART CHECKPOINT
WRITE CORE ON SAVE DEVICE
WRITE RESTART RECORD ON FILE
RETURN EVENT TO STORAGE
1
A

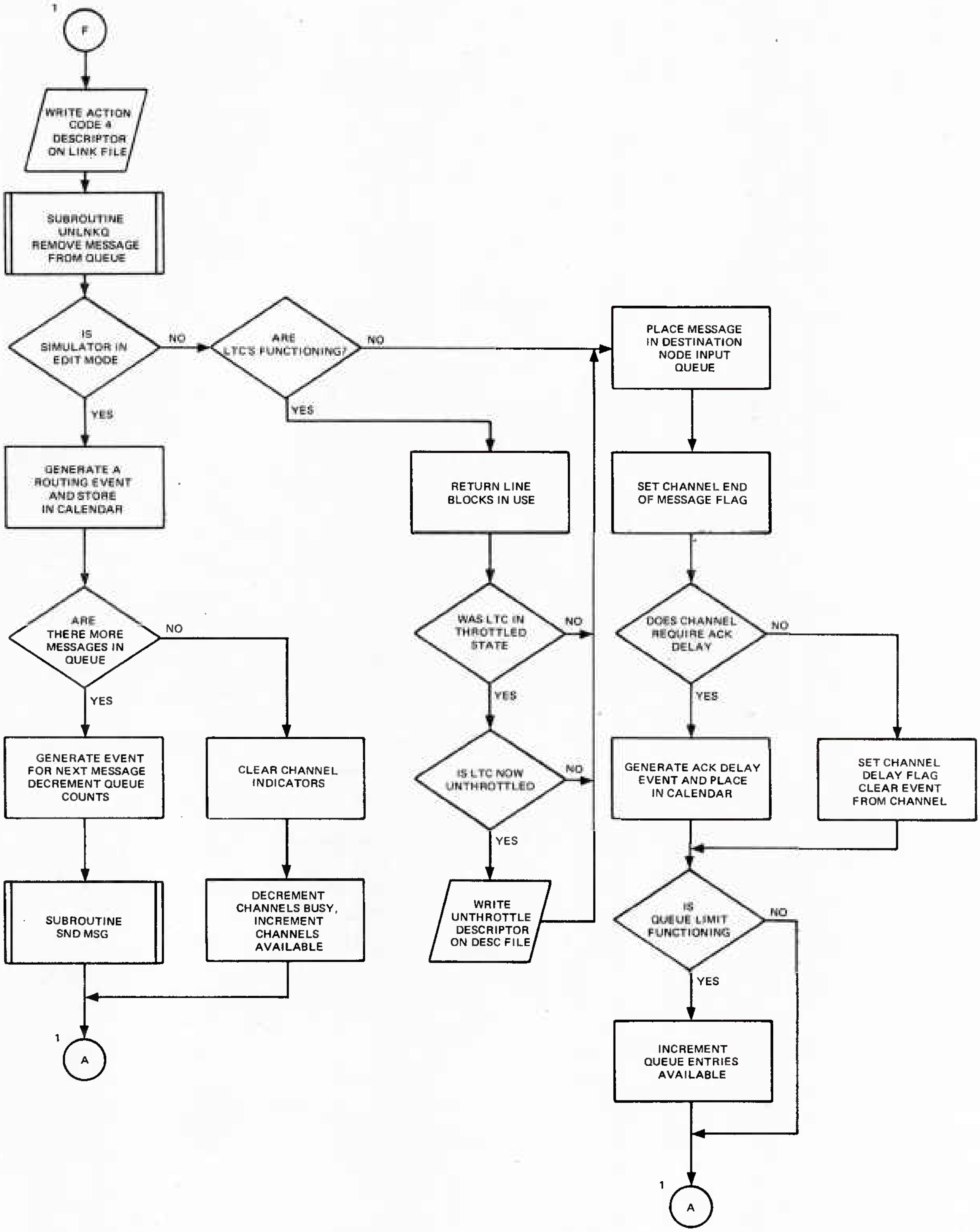
Basic Simulation Main Program (Sheet 3 of 9)

MESSAGE PROCESSOR EVENT (TYPE 2)



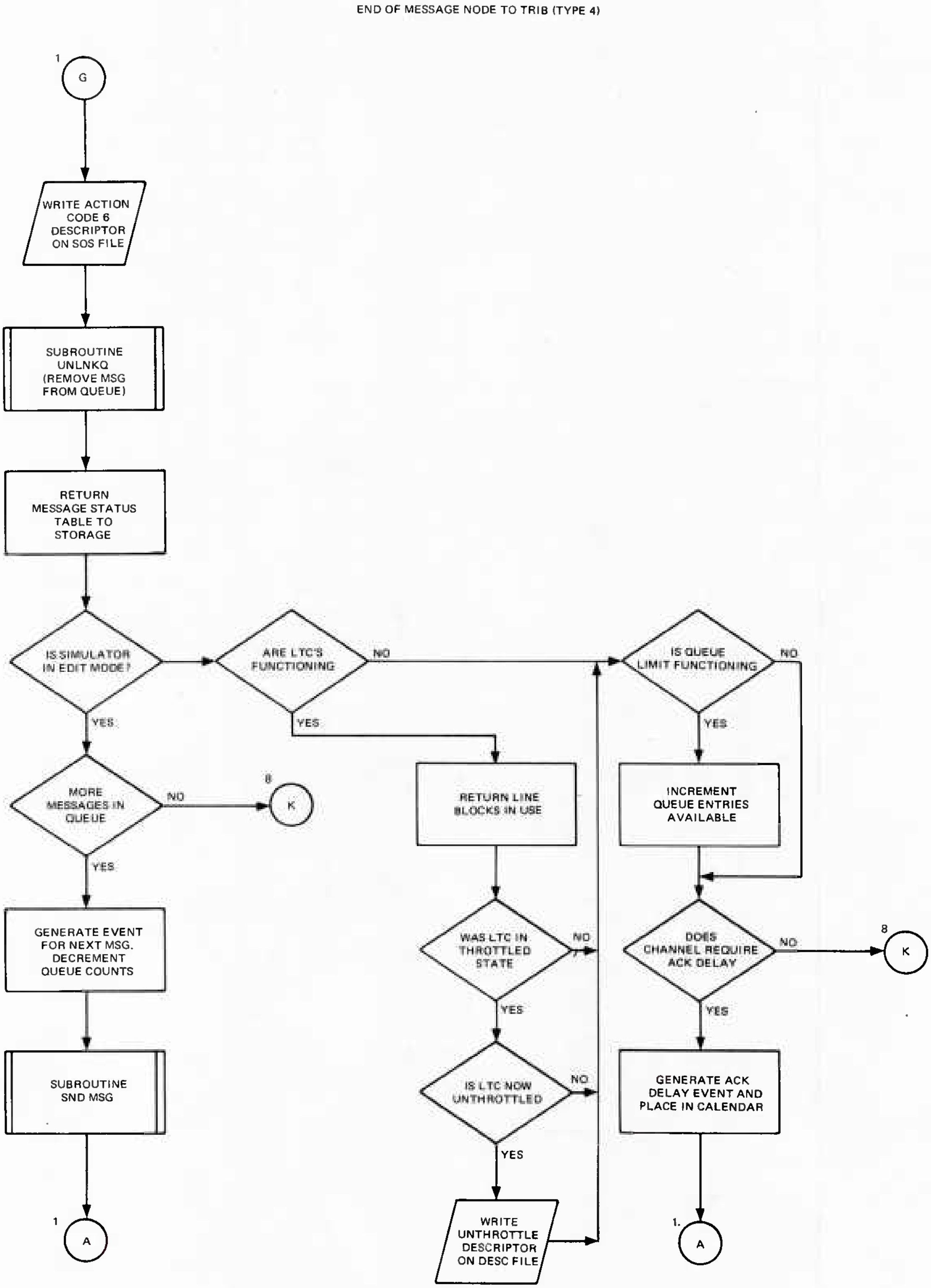
Basic Simulation Main Program (Sheet 4 of 9)

END OF MESSAGE NODE TO NODE EVENT (TYPE 3)



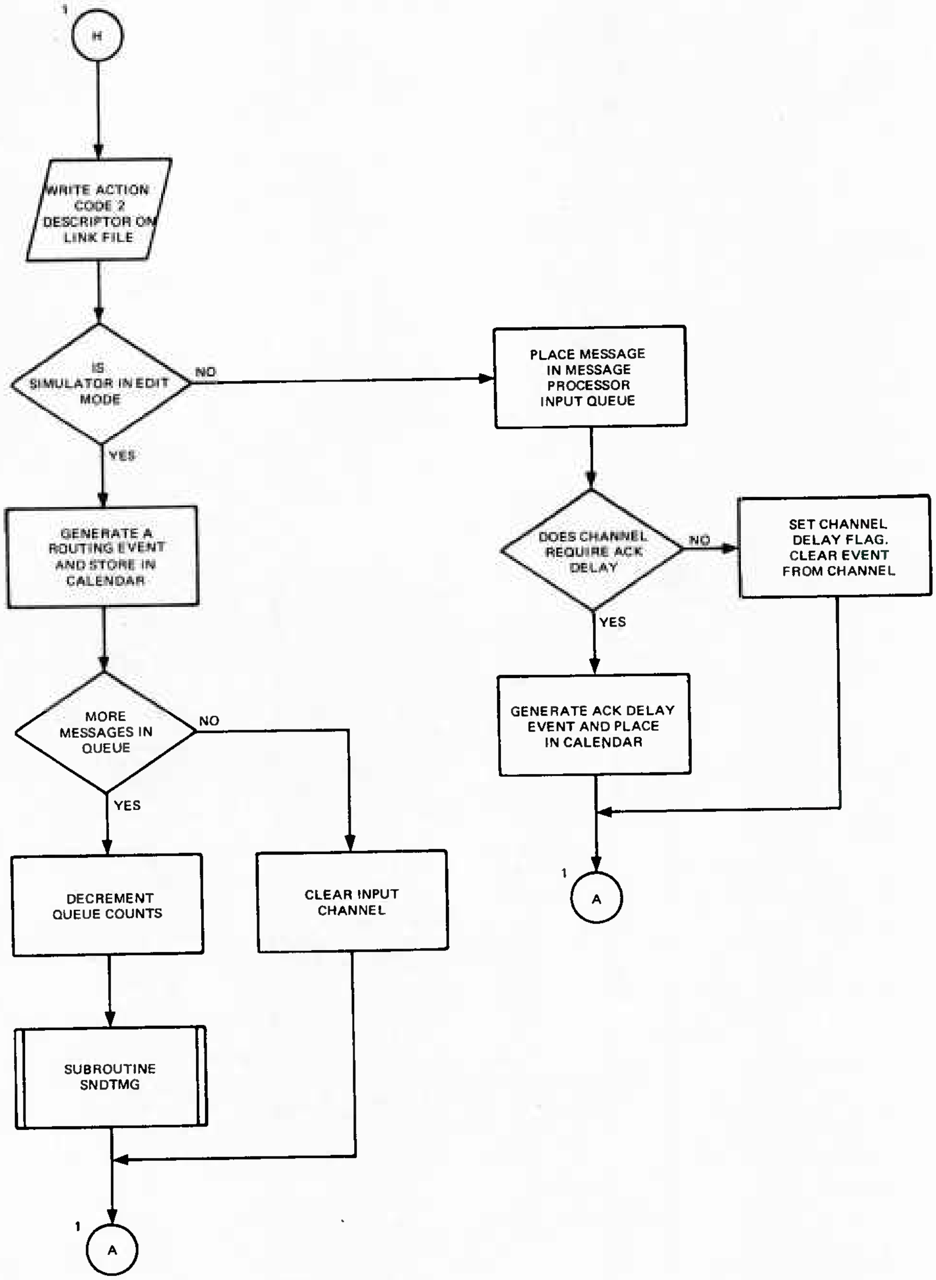
1
F
WRITE ACTION CODE 4 DESCRIPTOR ON LINK FILE
SUBROUTINE UNLNKQ REMOVE MESSAGE FROM QUEUE
IS SIMULATOR IN EDIT MODE
NO
YES
ARE LTC'S FUNCTIONING?
NO
YES
PLACE MESSAGE IN DESTINATION NODE INPUT QUEUE
GENERATE A ROUTING EVENT AND STORE IN CALENDAR
RETURN LINE BLOCKS IN USE
SET CHANNEL END OF MESSAGE FLAG
ARE THERE MORE MESSAGES IN QUEUE
NO
YES
WAS LTC IN THROTTLED STATE
NO
YES
DOES CHANNEL REQUIRE ACK DELAY
NO
YES
GENERATE EVENT FOR NEXT MESSAGE DECREMENT QUEUE COUNTS
CLEAR CHANNEL INDICATORS
IS LTC NOW UNTHROTTLED
NO
YES
GENERATE ACK DELAY EVENT AND PLACE IN CALENDAR
SET CHANNEL DELAY FLAG CLEAR EVENT FROM CHANNEL
SUBROUTINE SND MSG
DECREMENT CHANNELS BUSY, INCREMENT CHANNELS AVAILABLE
WRITE UNTHROTTLE DESCRIPTOR ON DESC FILE
IS QUEUE LIMIT FUNCTIONING
NO
YES
1
A
INCREMENT QUEUE ENTRIES AVAILABLE
1
A

Basic Simulation Main Program (Sheet 5 of 9)

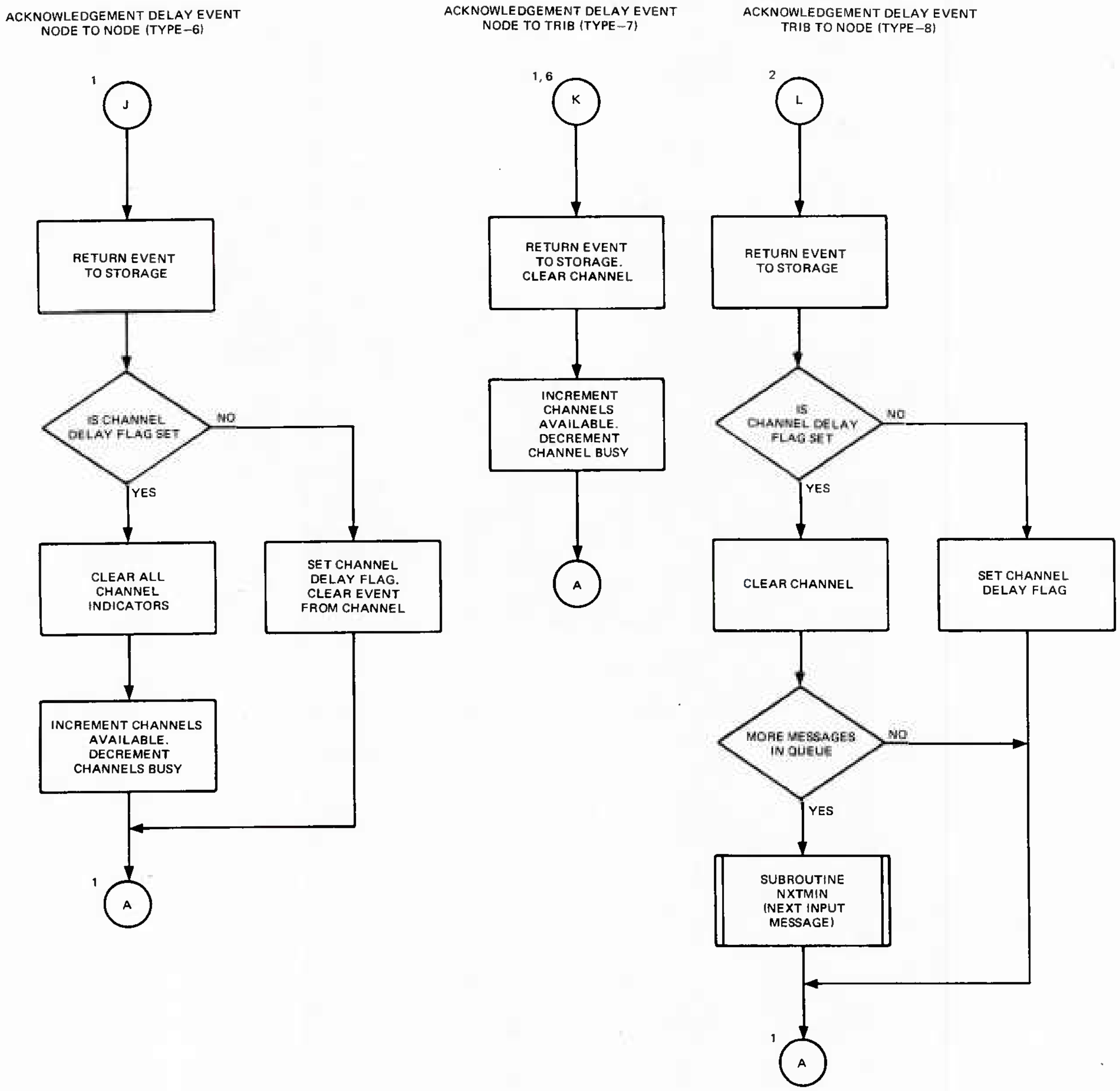

END OF MESSAGE NODE TO TRIB (TYPE 4)
1
G
WRITE ACTION CODE 6 DESCRIPTOR ON SOS FILE
SUBROUTINE UNLNKQ (REMOVE MSG FROM QUEUE)
RETURN MESSAGE STATUS TABLE TO STORAGE
IS SIMULATOR IN EDIT MODE?
YES
ARE LTC'S FUNCTIONING
NO
YES
IS QUEUE LIMIT FUNCTIONING
NO
YES
MORE MESSAGES IN QUEUE
NO
8
K
YES
RETURN LINE BLOCKS IN USE
INCREMENT QUEUE ENTRIES AVAILABLE
GENERATE EVENT FOR NEXT MSG. DECREMENT QUEUE COUNTS
WAS LTC IN THROTTLED STATE
NO
YES
DOES CHANNEL REQUIRE ACK DELAY
NO
8
K
YES
SUBROUTINE SND MSG
IS LTC NOW UNTHROTTLED
NO
YES
GENERATE ACK DELAY EVENT AND PLACE IN CALENDAR
1
A
WRITE UNTHROTTLE DESCRIPTOR ON DESC FILE
1.
A

Basic Simulation Main Program (Sheet 6 of 9)

END OF MESSAGE TRIB TO NODE (TYPE 5)

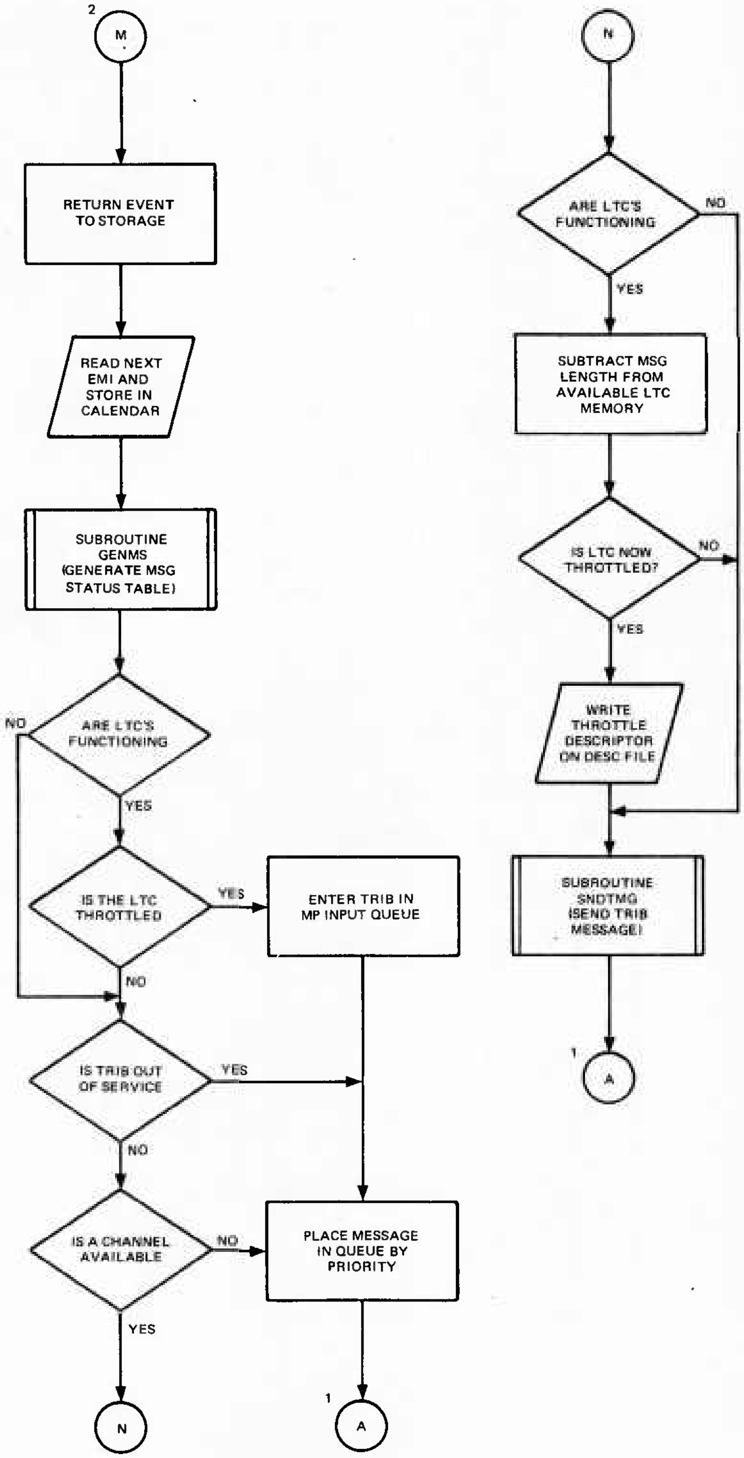

1
H
WRITE ACTION CODE 2 DESCRIPTOR ON LINK FILE
IS SIMULATOR IN EDIT MODE
NO
YES
PLACE MESSAGE IN MESSAGE PROCESSOR INPUT QUEUE
GENERATE A ROUTING EVENT AND STORE IN CALENDAR
DOES CHANNEL REQUIRE ACK DELAY
NO
SET CHANNEL DELAY FLAG. CLEAR EVENT FROM CHANNEL
YES
GENERATE ACK DELAY EVENT AND PLACE IN CALENDAR
MORE MESSAGES IN QUEUE
NO
YES
1
A
DECREMENT QUEUE COUNTS
CLEAR INPUT CHANNEL
SUBROUTINE SNDTMG
1
A

Basic Simulation Main Program (Sheet 7 of 9)

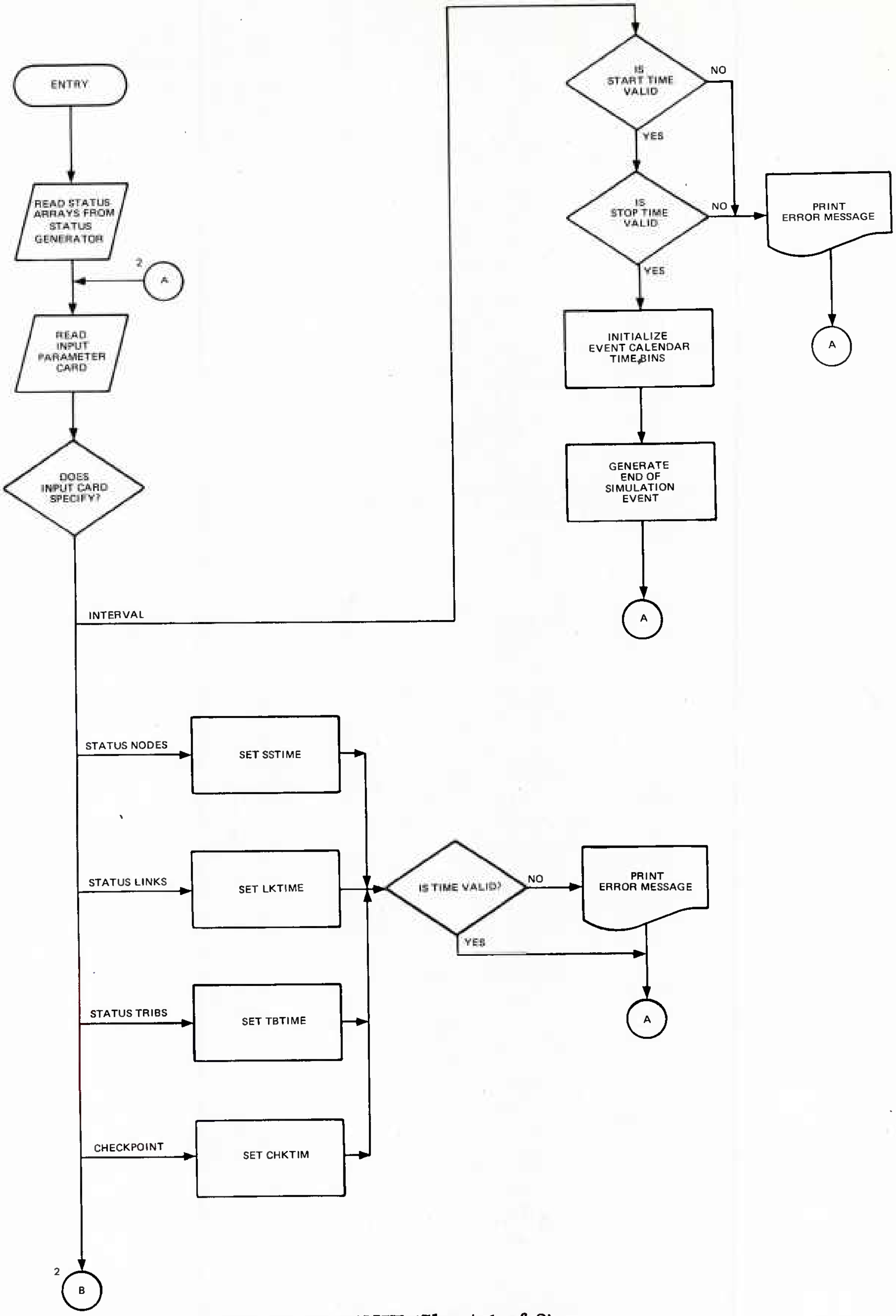

ACKNOWLEDGEMENT DELAY EVENT NODE TO NODE (TYPE–6)
1
J
RETURN EVENT TO STORAGE
IS CHANNEL DELAY FLAG SET
NO
YES
CLEAR ALL CHANNEL INDICATORS
SET CHANNEL DELAY FLAG. CLEAR EVENT FROM CHANNEL
INCREMENT CHANNELS AVAILABLE. DECREMENT CHANNELS BUSY
1
A
ACKNOWLEDGEMENT DELAY EVENT NODE TO TRIB (TYPE–7)
1, 6
K
RETURN EVENT TO STORAGE. CLEAR CHANNEL
INCREMENT CHANNELS AVAILABLE. DECREMENT CHANNEL BUSY
A
ACKNOWLEDGEMENT DELAY EVENT TRIB TO NODE (TYPE–8)
2
L
RETURN EVENT TO STORAGE
IS CHANNEL DELAY FLAG SET
NO
YES
CLEAR CHANNEL
SET CHANNEL DELAY FLAG
MORE MESSAGES IN QUEUE
NO
YES
SUBROUTINE NXTMIN (NEXT INPUT MESSAGE)
1
A

Basic Simulation Main Program (Sheet 8 of 9)

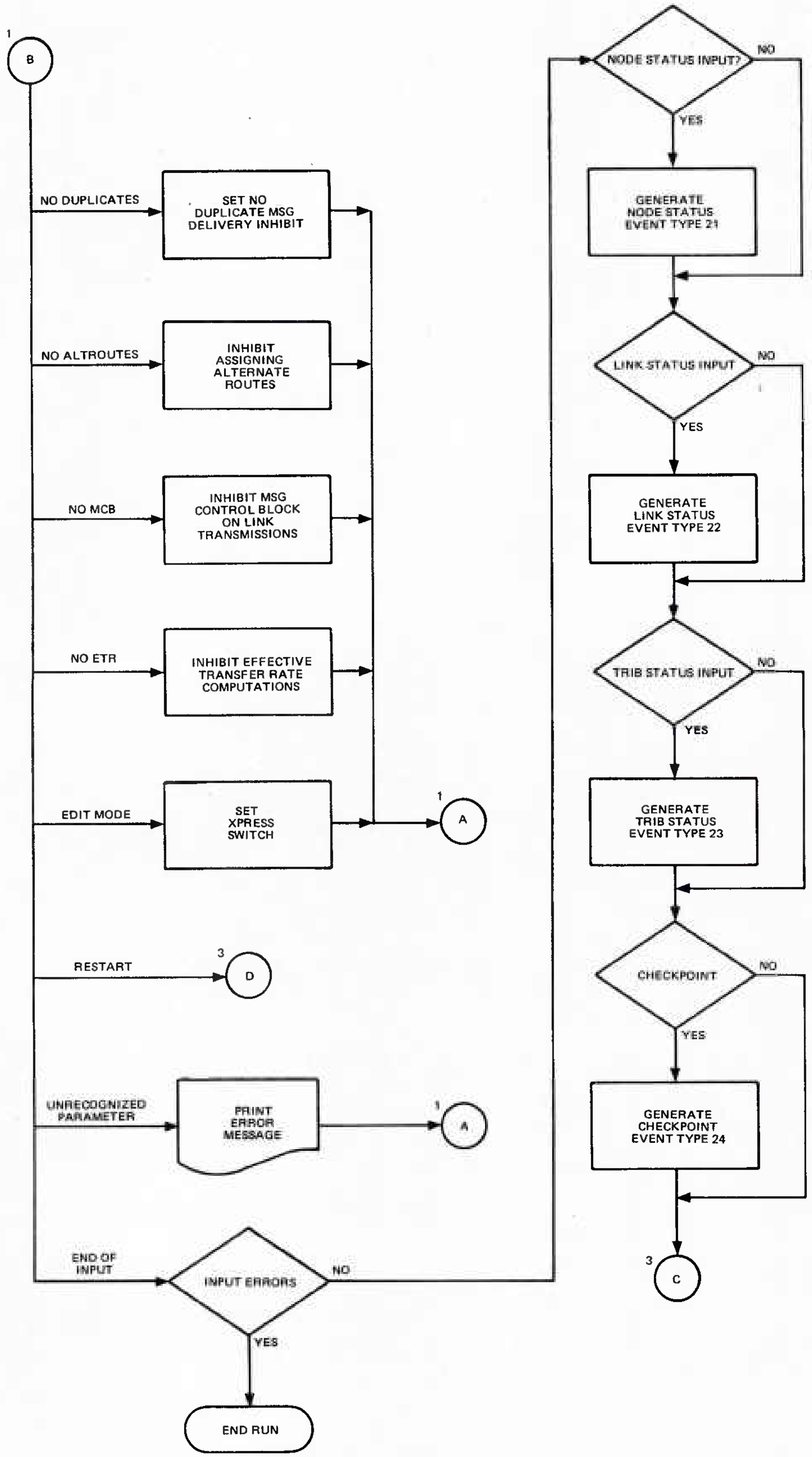
MESSAGE INPUT EVENT (TYPE 11)


2
M
RETURN EVENT TO STORAGE
READ NEXT EMI AND STORE IN CALENDAR
SUBROUTINE GENMS (GENERATE MSG STATUS TABLE)
ARE LTC'S FUNCTIONING
NO
YES
IS THE LTC THROTTLED
YES
ENTER TRIB IN MP INPUT QUEUE
NO
IS TRIB OUT OF SERVICE
YES
NO
IS A CHANNEL AVAILABLE
NO
PLACE MESSAGE IN QUEUE BY PRIORITY
YES
N
1
A
N
ARE LTC'S FUNCTIONING
NO
YES
SUBTRACT MSG LENGTH FROM AVAILABLE LTC MEMORY
IS LTC NOW THROTTLED?
NO
YES
WRITE THROTTLE DESCRIPTOR ON DESC FILE
SUBROUTINE SNDTMG (SEND TRIB MESSAGE)
1
A

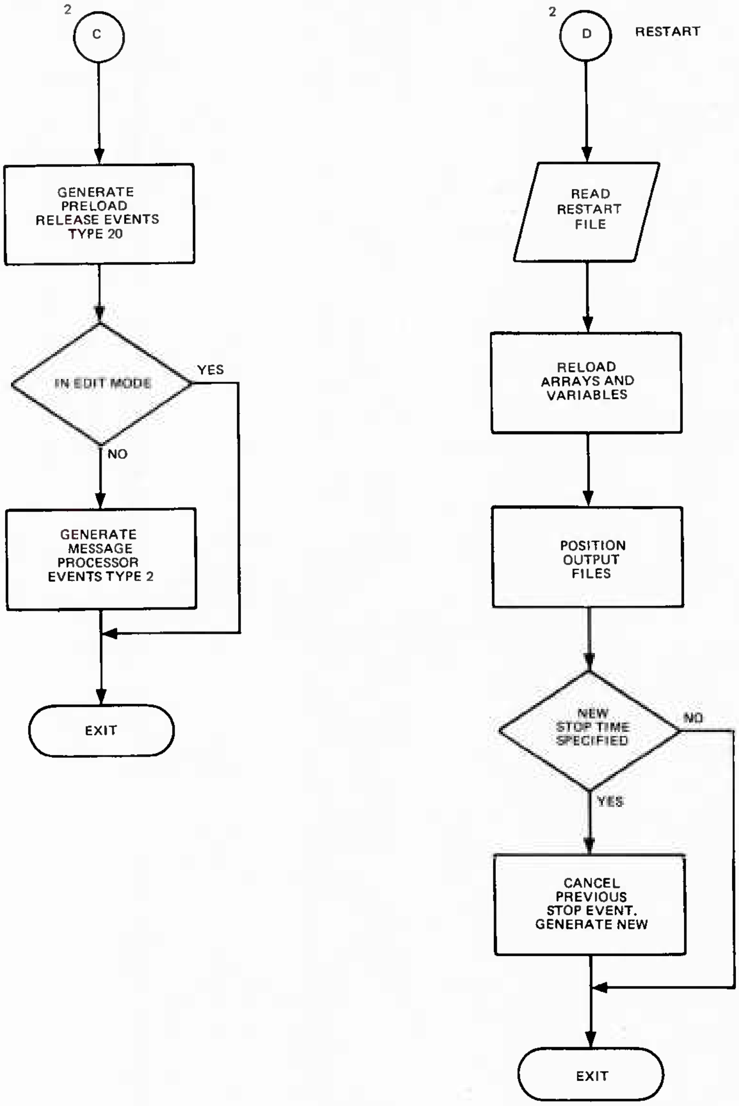
Basic Simulation Main Program (Sheet 9 of 9)



Subroutine INIT (Sheet 1 of 3)

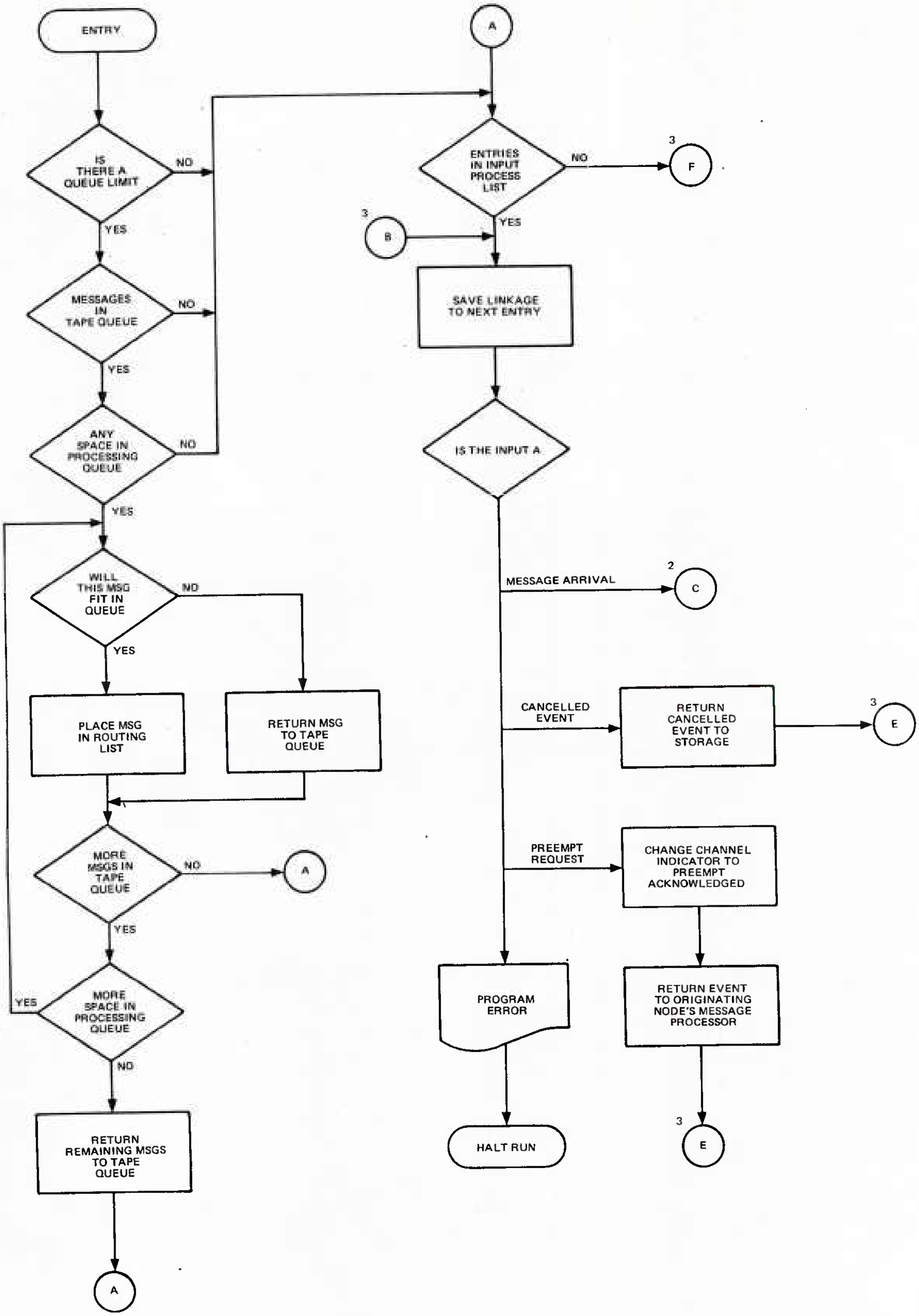

1
B
NO DUPLICATES
SET NO DUPLICATE MSG DELIVERY INHIBIT
NO ALTROUTES
INHIBIT ASSIGNING ALTERNATE ROUTES
NO MCB
INHIBIT MSG CONTROL BLOCK ON LINK TRANSMISSIONS
NO ETR
INHIBIT EFFECTIVE TRANSFER RATE COMPUTATIONS
EDIT MODE
SET XPRESS SWITCH
1
A
RESTART
3
D
UNRECOGNIZED PARAMETER
PRINT ERROR MESSAGE
1
A
END OF INPUT
INPUT ERRORS
NO
YES
END RUN
NODE STATUS INPUT?
NO
YES
GENERATE NODE STATUS EVENT TYPE 21
LINK STATUS INPUT
NO
YES
GENERATE LINK STATUS EVENT TYPE 22
TRIB STATUS INPUT
NO
YES
GENERATE TRIB STATUS EVENT TYPE 23
CHECKPOINT
NO
YES
GENERATE CHECKPOINT EVENT TYPE 24
3
C

Subroutine INIT (Sheet 2 of 3)



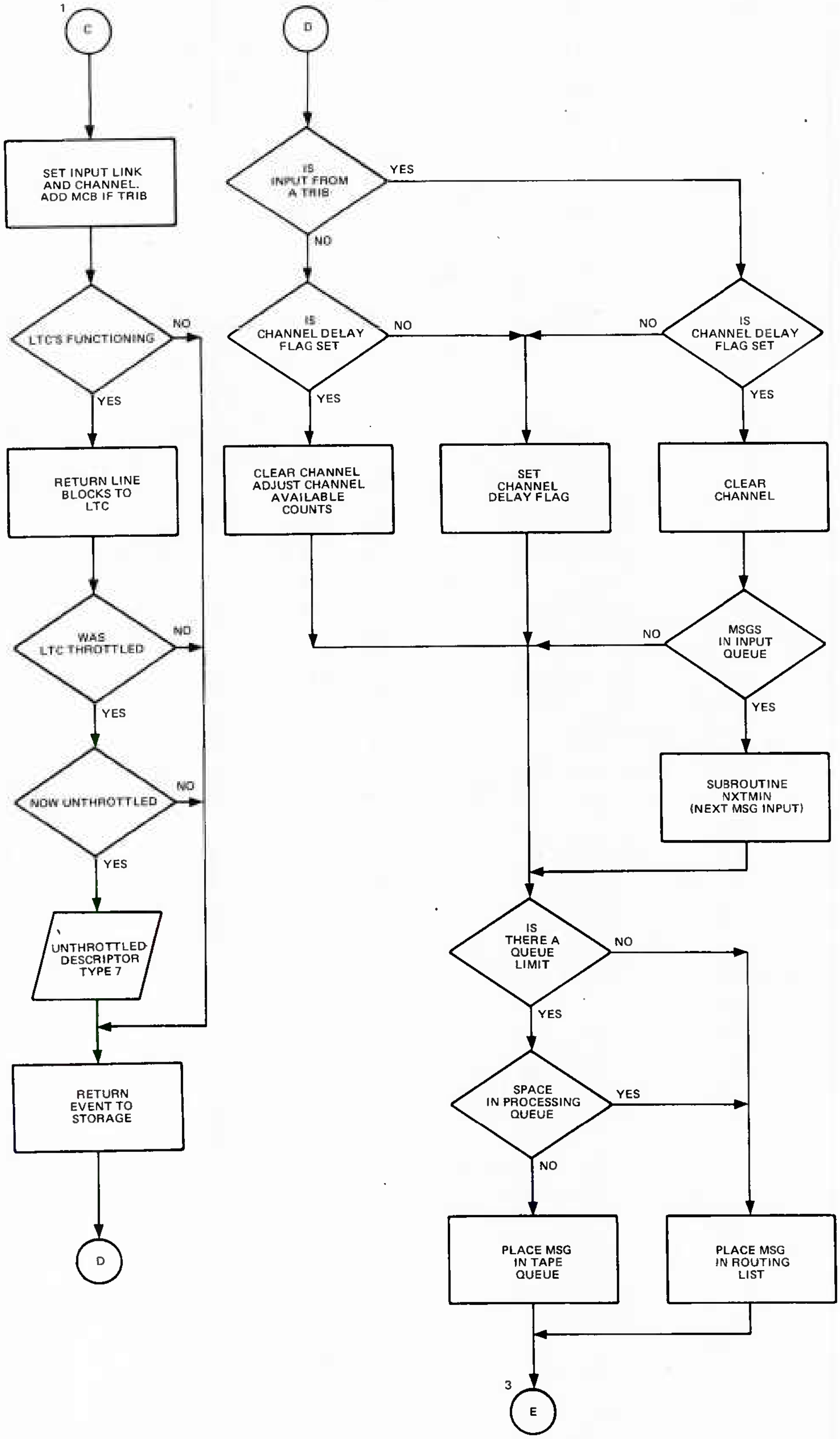
2
C
GENERATE PRELOAD RELEASE EVENTS TYPE 20
IN EDIT MODE
YES
NO
GENERATE MESSAGE PROCESSOR EVENTS TYPE 2
EXIT
2
D
RESTART
READ RESTART FILE
RELOAD ARRAYS AND VARIABLES
POSITION OUTPUT FILES
NEW STOP TIME SPECIFIED
NO
YES
CANCEL PREVIOUS STOP EVENT. GENERATE NEW
EXIT

Subroutine INIT (Sheet 3 of 3)

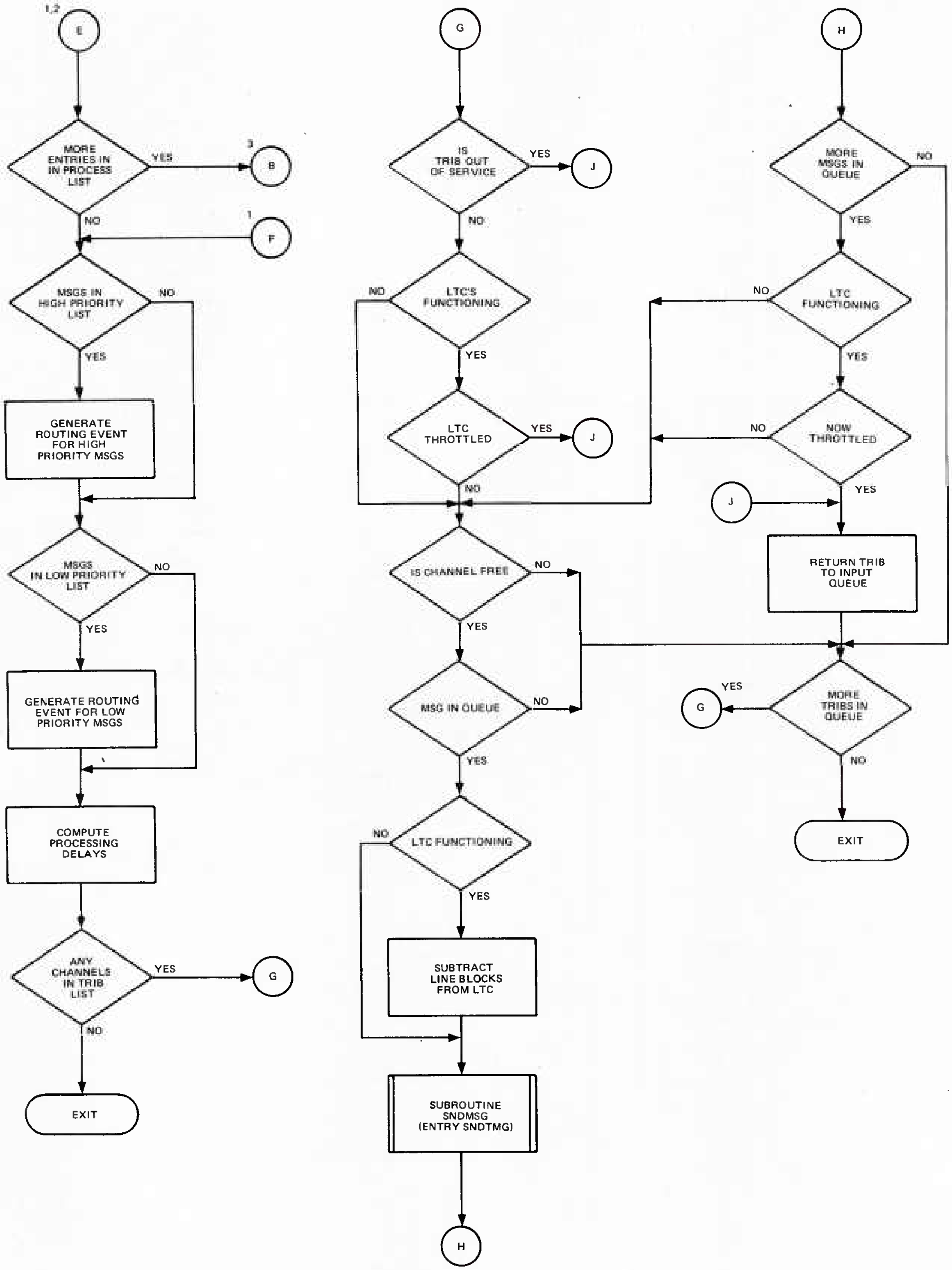


ENTRY
IS THERE A QUEUE LIMIT
NO
YES
MESSAGES IN TAPE QUEUE
NO
YES
ANY SPACE IN PROCESSING QUEUE
NO
YES
WILL THIS MSG FIT IN QUEUE
NO
YES
PLACE MSG IN ROUTING LIST
RETURN MSG TO TAPE QUEUE
MORE MSGS IN TAPE QUEUE
NO
A
YES
YES
MORE SPACE IN PROCESSING QUEUE
NO
RETURN REMAINING MSGS TO TAPE QUEUE
A
A
ENTRIES IN INPUT PROCESS LIST
NO
3
F
YES
3
B
SAVE LINKAGE TO NEXT ENTRY
IS THE INPUT A
MESSAGE ARRIVAL
2
C
CANCELLED EVENT
RETURN CANCELLED EVENT TO STORAGE
3
E
PREEMPT REQUEST
CHANGE CHANNEL INDICATOR TO PREEMPT ACKNOWLEDGED
PROGRAM ERROR
RETURN EVENT TO ORIGINATING NODE'S MESSAGE PROCESSOR
HALT RUN
3
E

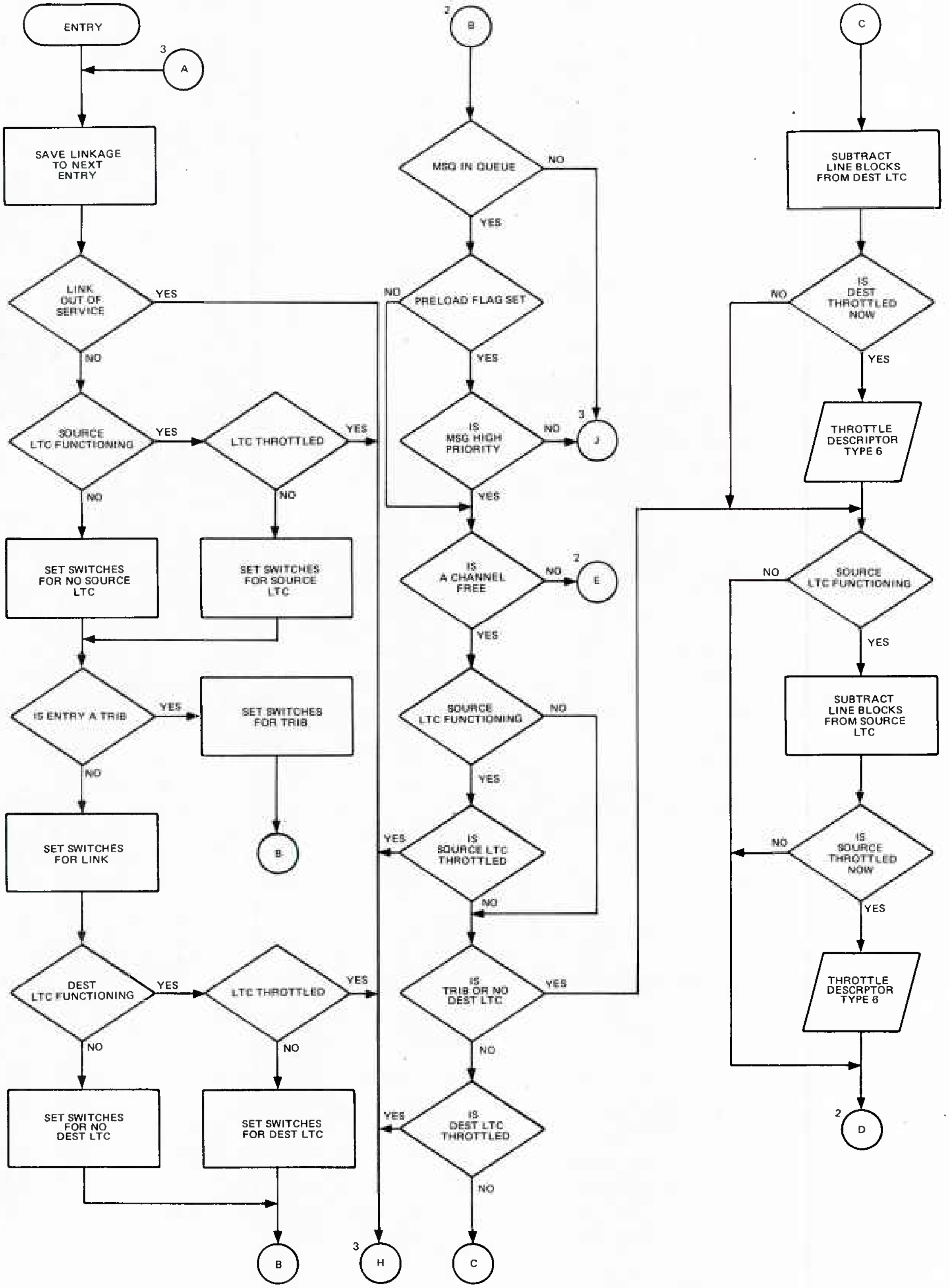
Subroutine PID (Sheet 1 of 3)


1
C
SET INPUT LINK AND CHANNEL. ADD MCB IF TRIB
LTC'S FUNCTIONING
NO
YES
RETURN LINE BLOCKS TO LTC
WAS LTC THROTTLED
NO
YES
NOW UNTHROTTLED
NO
YES
UNTHROTTLED DESCRIPTOR TYPE 7
RETURN EVENT TO STORAGE
D
D
IS INPUT FROM A TRIB
YES
NO
IS CHANNEL DELAY FLAG SET
NO
YES
CLEAR CHANNEL ADJUST CHANNEL AVAILABLE COUNTS
SET CHANNEL DELAY FLAG
IS CHANNEL DELAY FLAG SET
NO
YES
CLEAR CHANNEL
MSGS IN INPUT QUEUE
NO
YES
SUBROUTINE NXTMIN (NEXT MSG INPUT)
IS THERE A QUEUE LIMIT
NO
YES
SPACE IN PROCESSING QUEUE
YES
NO
PLACE MSG IN TAPE QUEUE
PLACE MSG IN ROUTING LIST
3
E

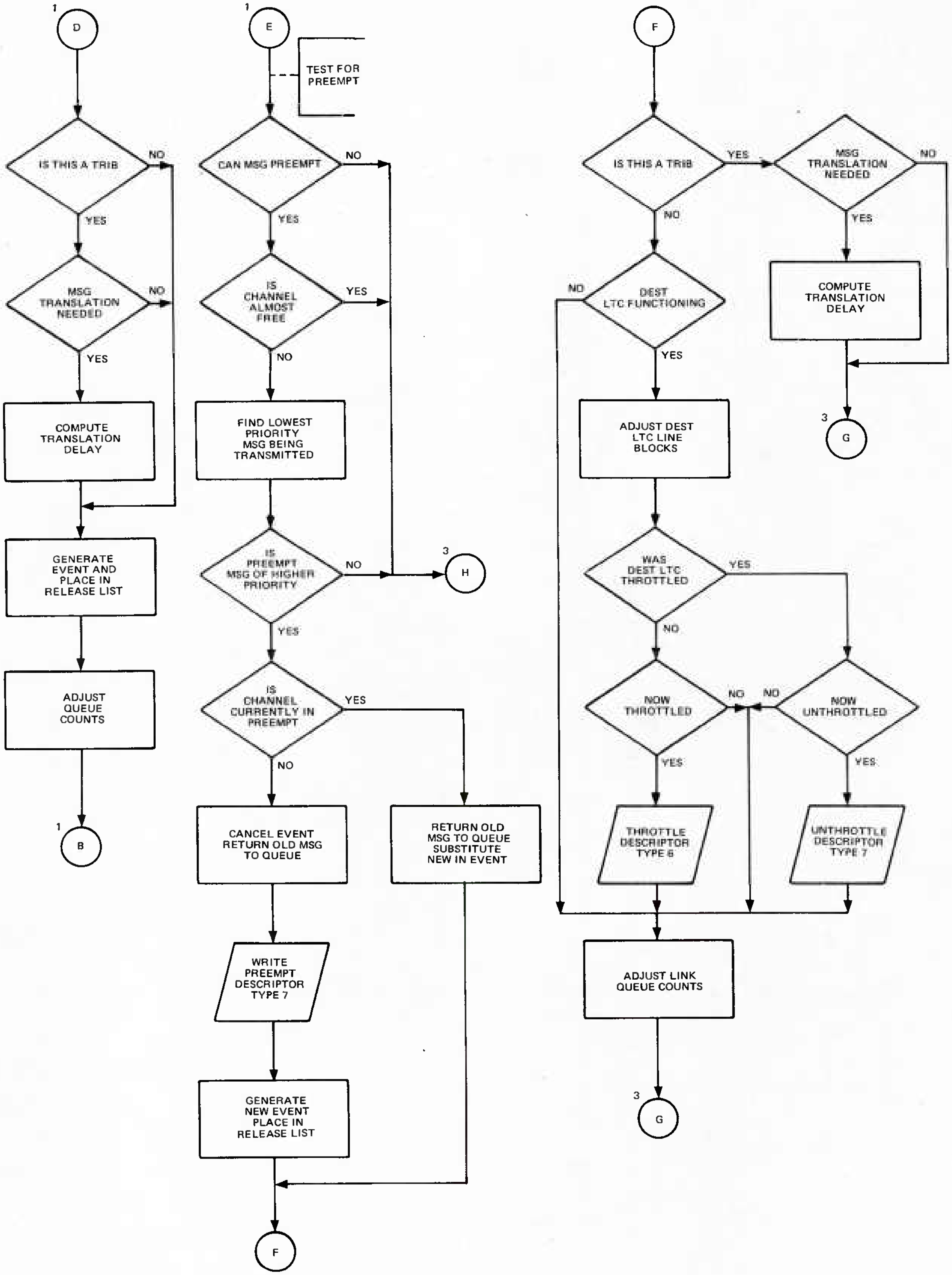
Subroutine PID (Sheet 2 of 3)


1,2
E
MORE ENTRIES IN IN PROCESS LIST
YES
3
B
NO
1
F
MSGS IN HIGH PRIORITY LIST
NO
YES
GENERATE ROUTING EVENT FOR HIGH PRIORITY MSGS
MSGS IN LOW PRIORITY LIST
NO
YES
GENERATE ROUTING EVENT FOR LOW PRIORITY MSGS
COMPUTE PROCESSING DELAYS
ANY CHANNELS IN TRIB LIST
YES
G
NO
EXIT
G
IS TRIB OUT OF SERVICE
YES
J
NO
LTC'S FUNCTIONING
NO
YES
LTC THROTTLED
YES
J
NO
IS CHANNEL FREE
NO
YES
MSG IN QUEUE
NO
YES
LTC FUNCTIONING
NO
YES
SUBTRACT LINE BLOCKS FROM LTC
SUBROUTINE SNDMSG (ENTRY SNDTMG)
H
H
MORE MSGS IN QUEUE
NO
YES
LTC FUNCTIONING
NO
YES
NOW THROTTLED
NO
YES
J
RETURN TRIB TO INPUT QUEUE
MORE TRIBS IN QUEUE
YES
G
NO
EXIT

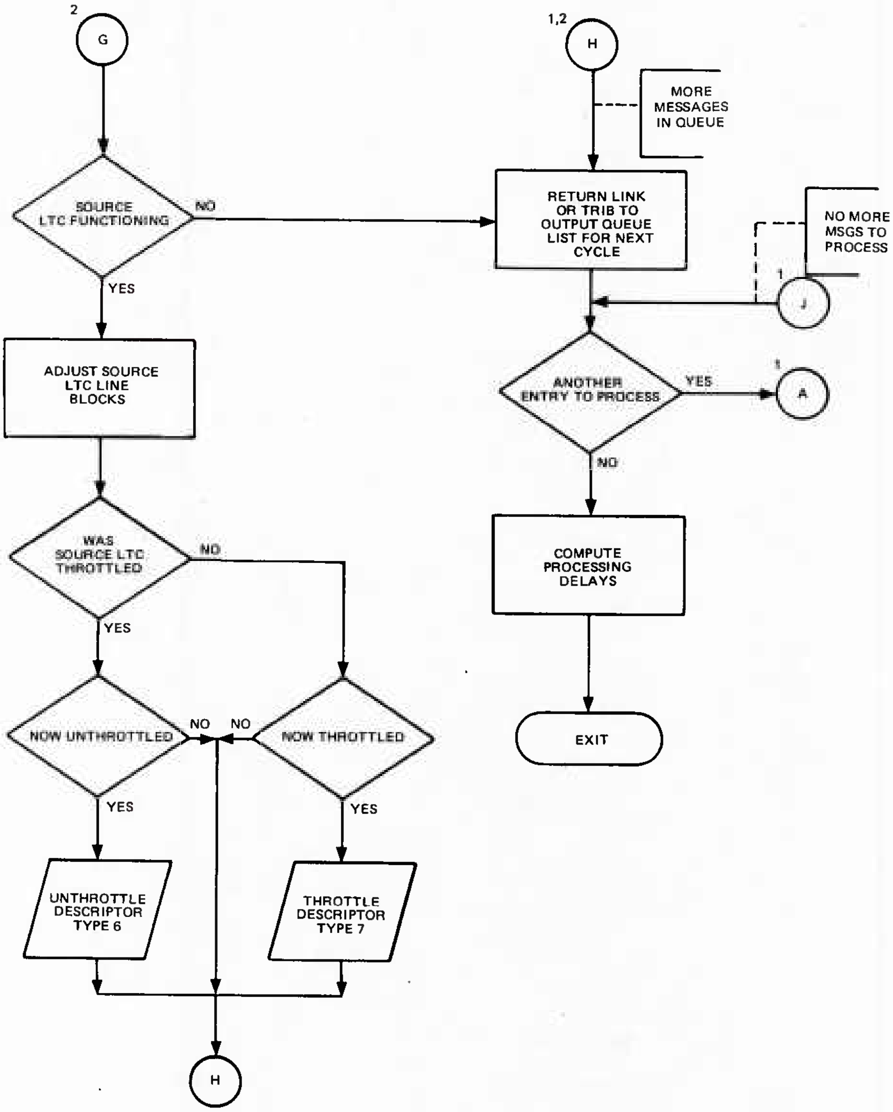
Subroutine PID (Sheet 3 of 3)



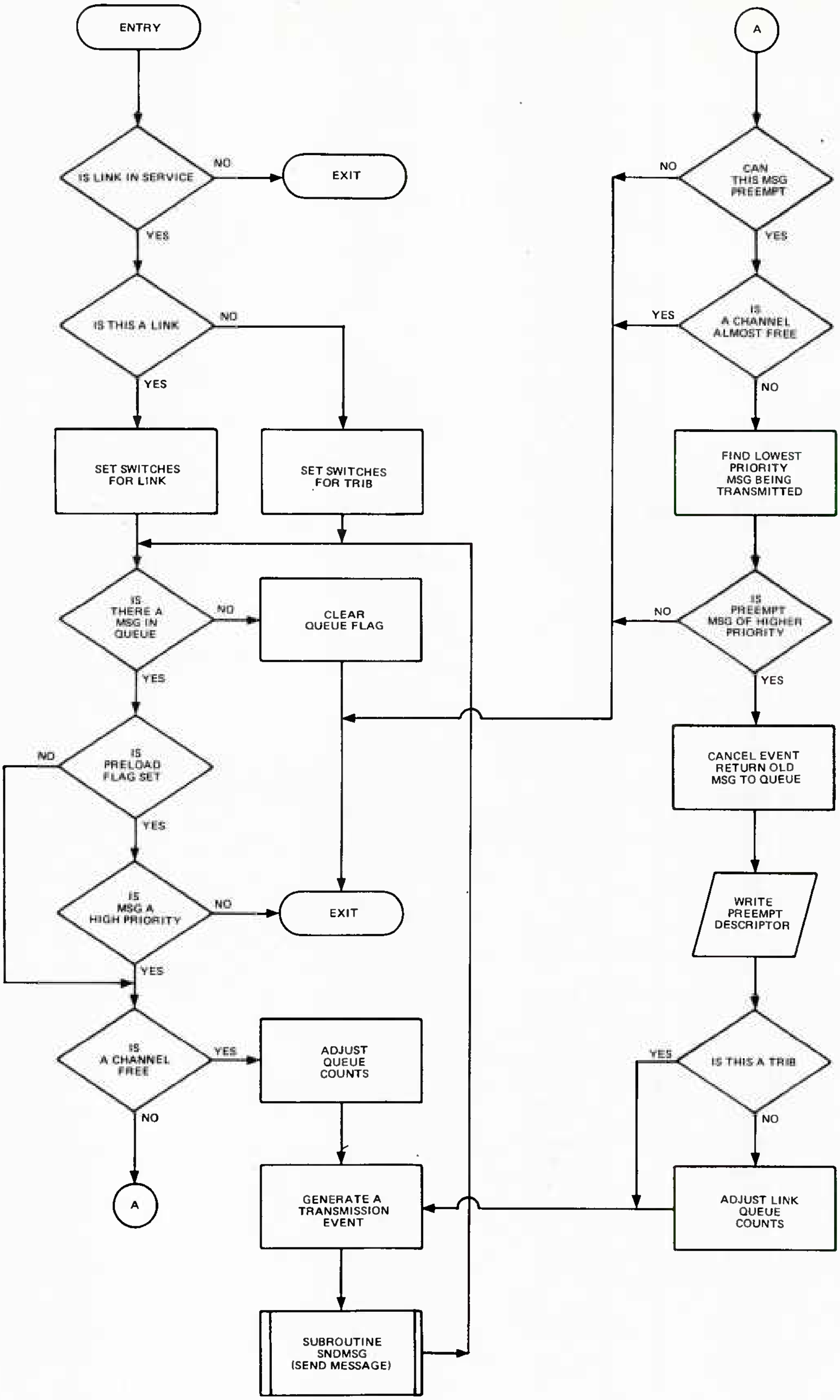
Subroutine POD (Sheet 1 of 3)


1
D
IS THIS A TRIB
NO
YES
MSG TRANSLATION NEEDED
NO
YES
COMPUTE TRANSLATION DELAY
GENERATE EVENT AND PLACE IN RELEASE LIST
ADJUST QUEUE COUNTS
1
B
1
E
TEST FOR PREEMPT
CAN MSG PREEMPT
NO
YES
IS CHANNEL ALMOST FREE
YES
NO
FIND LOWEST PRIORITY MSG BEING TRANSMITTED
IS PREEMPT MSG OF HIGHER PRIORITY
NO
3
H
YES
IS CHANNEL CURRENTLY IN PREEMPT
YES
NO
CANCEL EVENT RETURN OLD MSG TO QUEUE
RETURN OLD MSG TO QUEUE SUBSTITUTE NEW IN EVENT
WRITE PREEMPT DESCRIPTOR TYPE 7
GENERATE NEW EVENT PLACE IN RELEASE LIST
F
F
IS THIS A TRIB
YES
NO
MSG TRANSLATION NEEDED
NO
YES
COMPUTE TRANSLATION DELAY
3
G
NO
DEST LTC FUNCTIONING
YES
ADJUST DEST LTC LINE BLOCKS
WAS DEST LTC THROTTLED
YES
NO
NOW THROTTLED
NO
NO
NOW UNTHROTTLED
YES
YES
THROTTLE DESCRIPTOR TYPE 6
UNTHROTTLE DESCRIPTOR TYPE 7
ADJUST LINK QUEUE COUNTS
3
G

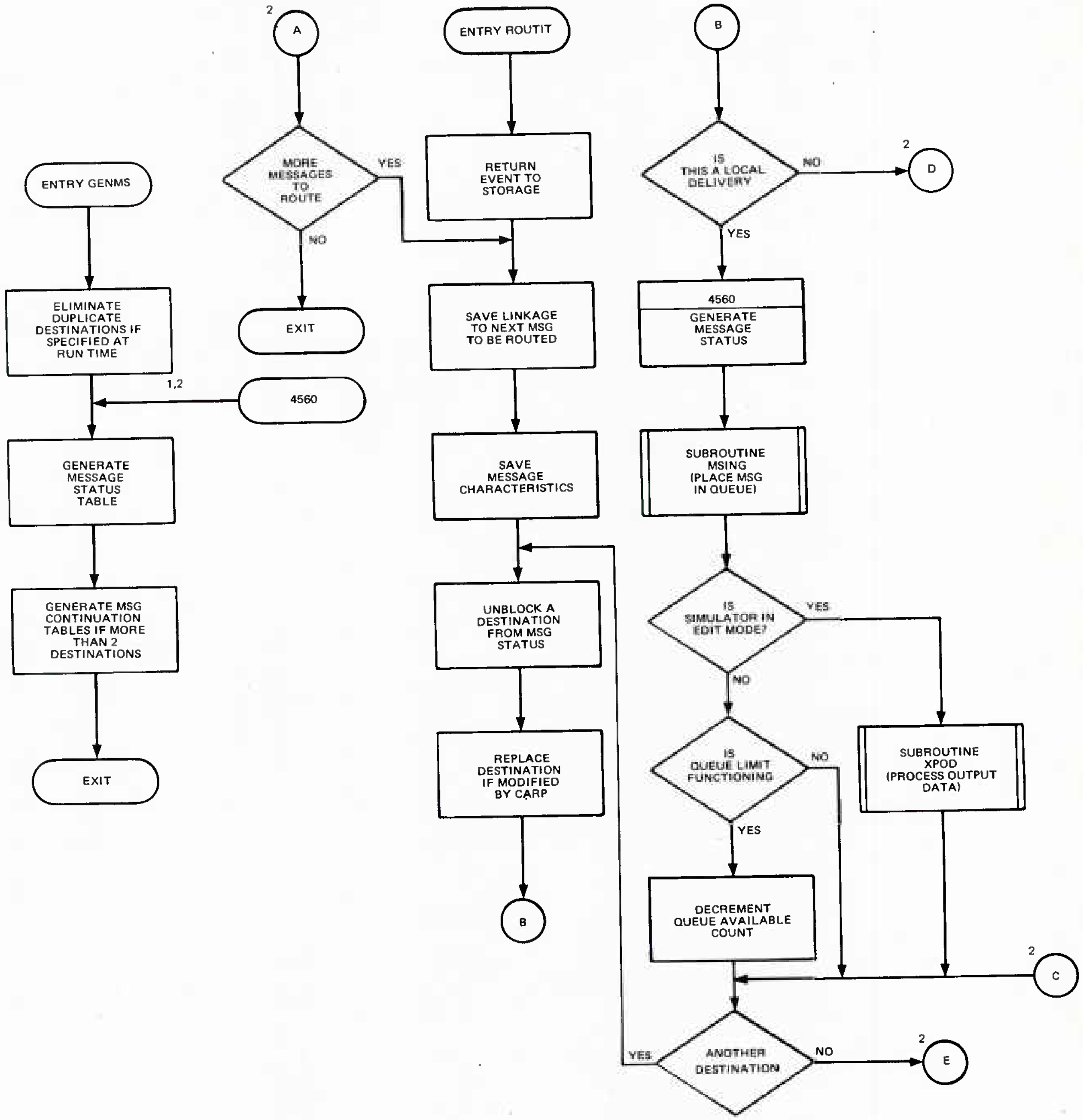
Subroutine POD (Sheet 2 of 3)



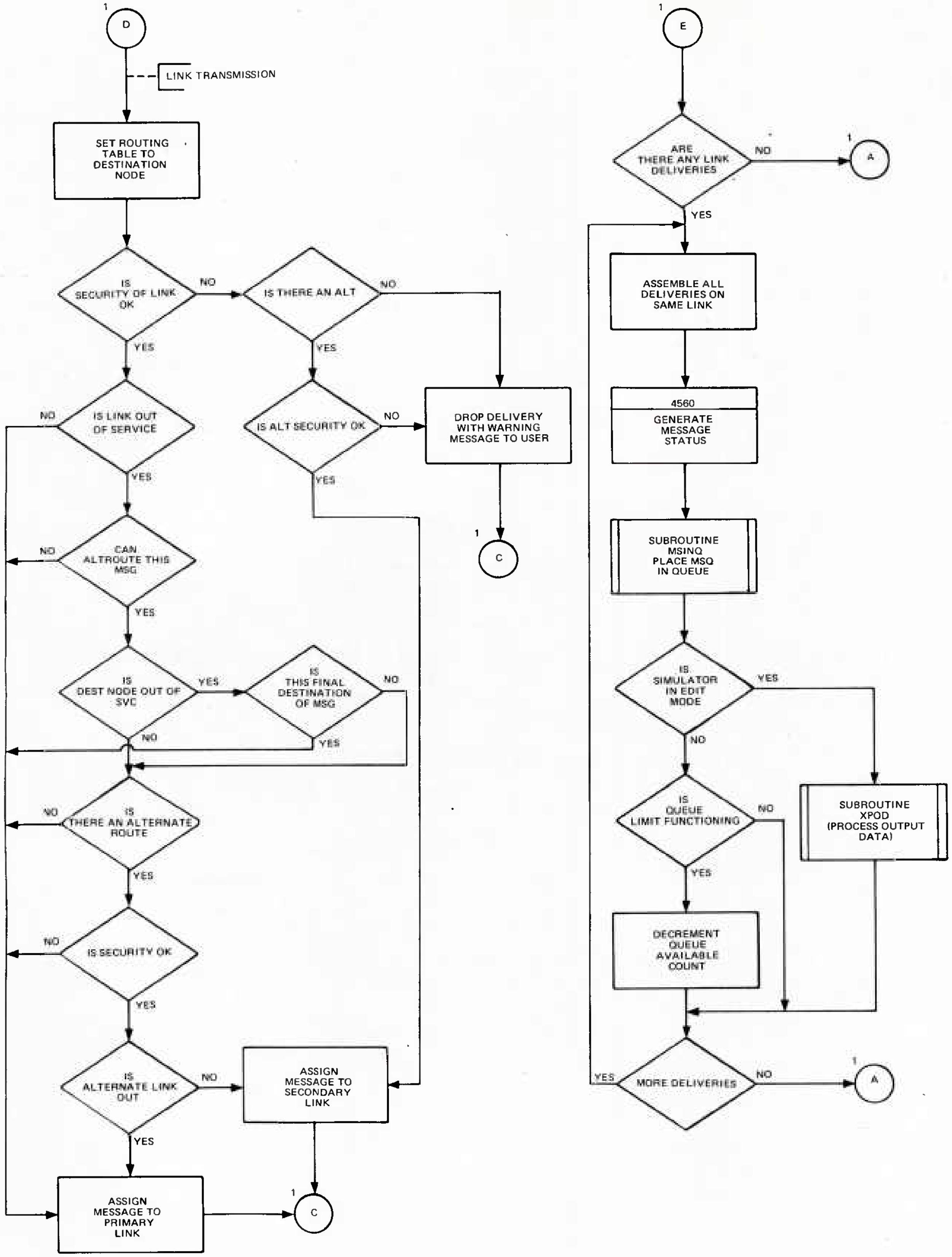
Subroutine POD (Sheet 3 of 3)



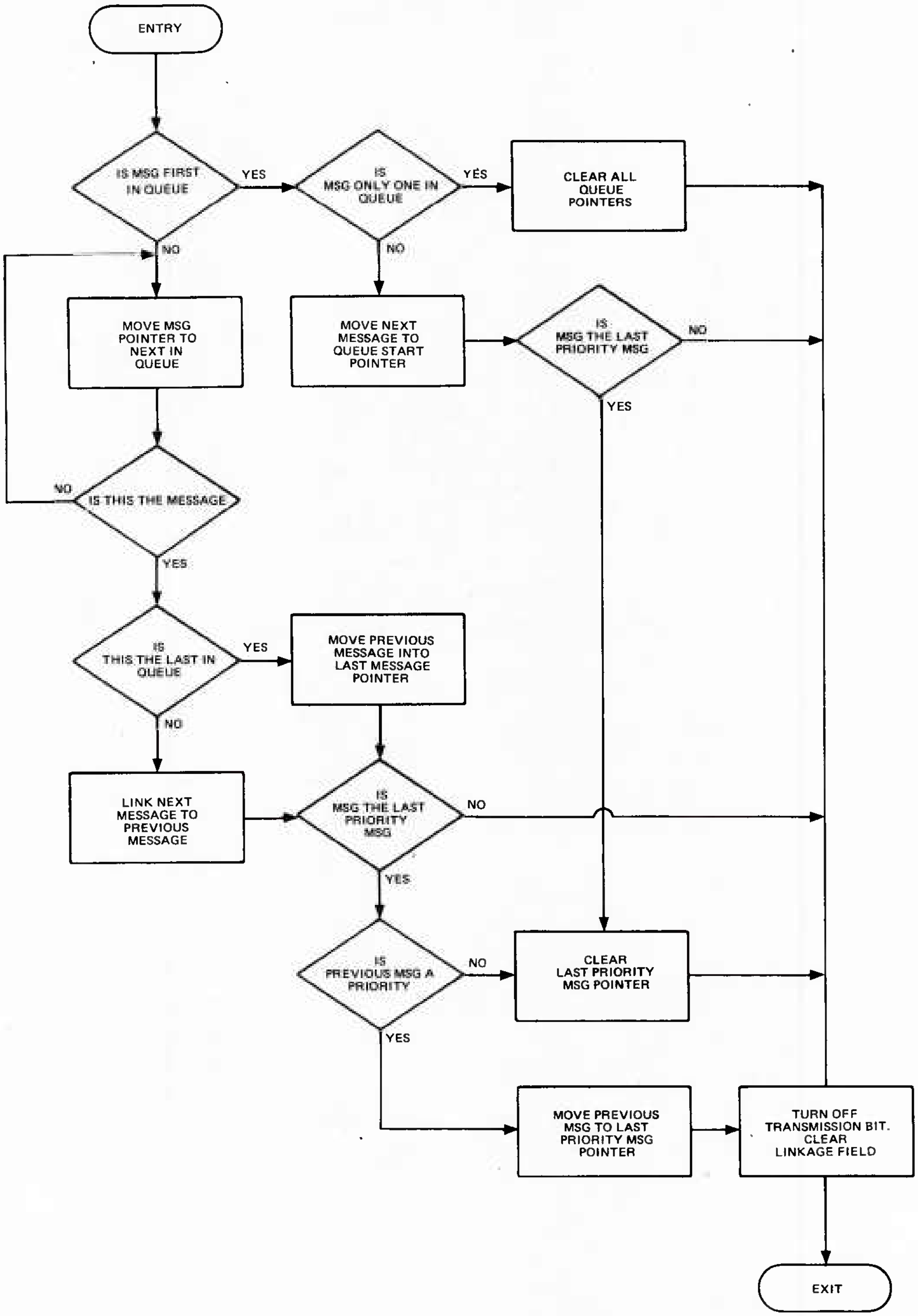
Subroutine XPOD (Sheet 1 of 1)


2
A
MORE MESSAGES TO ROUTE
YES
NO
EXIT
ENTRY ROUTIT
RETURN EVENT TO STORAGE
SAVE LINKAGE TO NEXT MSG TO BE ROUTED
SAVE MESSAGE CHARACTERISTICS
UNBLOCK A DESTINATION FROM MSG STATUS
REPLACE DESTINATION IF MODIFIED BY CARP
B
B
IS THIS A LOCAL DELIVERY
NO
2
D
YES
4560
GENERATE MESSAGE STATUS
SUBROUTINE MSING (PLACE MSG IN QUEUE)
IS SIMULATOR IN EDIT MODE?
YES
NO
SUBROUTINE XPOD (PROCESS OUTPUT DATA)
IS QUEUE LIMIT FUNCTIONING
NO
YES
DECREMENT QUEUE AVAILABLE COUNT
2
C
YES
ANOTHER DESTINATION
NO
2
E
ENTRY GENMS
ELIMINATE DUPLICATE DESTINATIONS IF SPECIFIED AT RUN TIME
1,2
4560
GENERATE MESSAGE STATUS TABLE
GENERATE MSG CONTINUATION TABLES IF MORE THAN 2 DESTINATIONS
EXIT

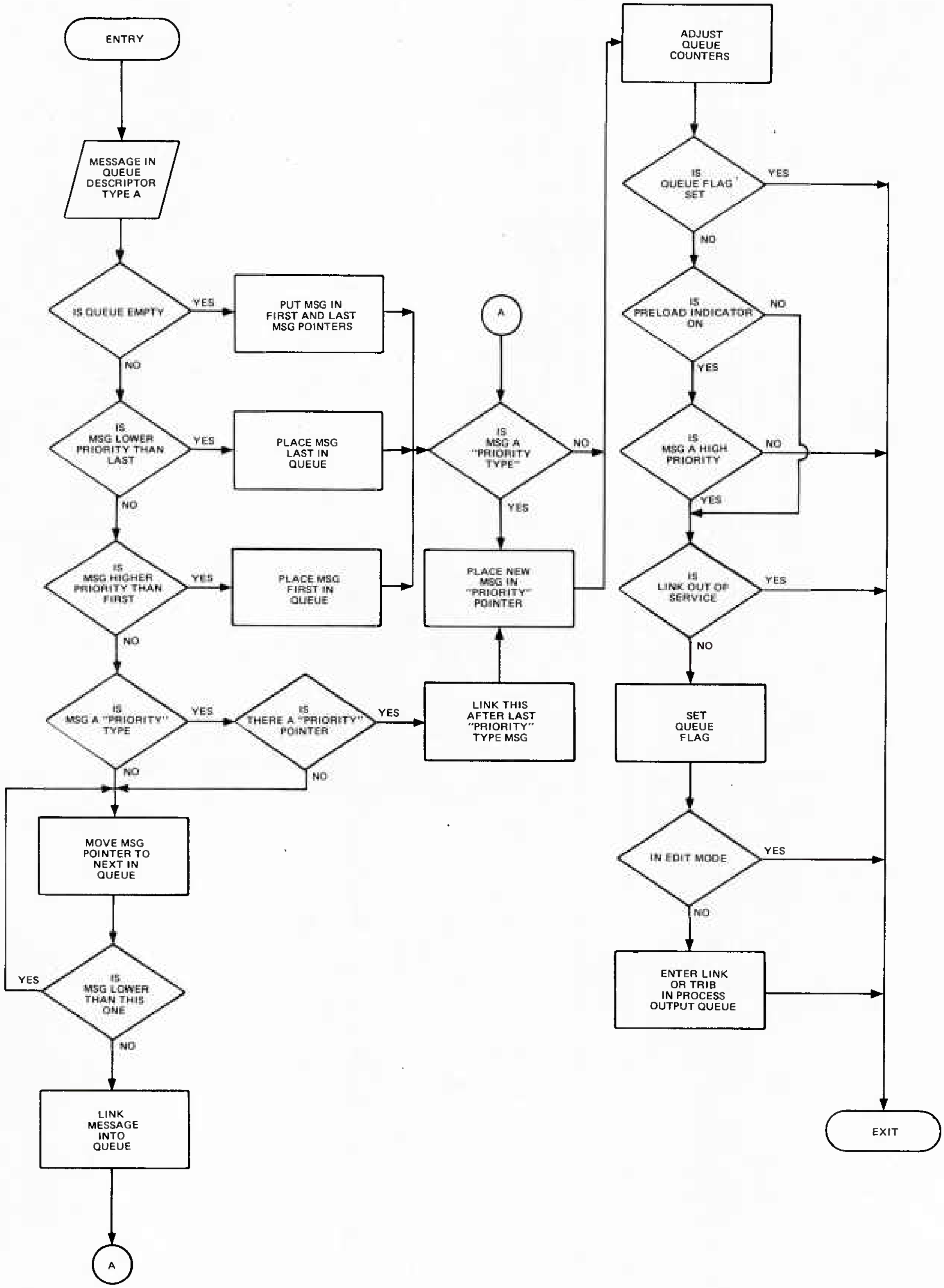
Subroutine GENMS (Sheet 1 of 2)


1
D
LINK TRANSMISSION
SET ROUTING TABLE TO DESTINATION NODE
IS SECURITY OF LINK OK
NO
YES
IS THERE AN ALT
NO
YES
IS LINK OUT OF SERVICE
NO
YES
IS ALT SECURITY OK
NO
YES
DROP DELIVERY WITH WARNING MESSAGE TO USER
1
C
CAN ALTROUTE THIS MSG
NO
YES
IS DEST NODE OUT OF SVC
YES
NO
IS THIS FINAL DESTINATION OF MSG
NO
YES
IS THERE AN ALTERNATE ROUTE
NO
YES
IS SECURITY OK
NO
YES
IS ALTERNATE LINK OUT
NO
YES
ASSIGN MESSAGE TO SECONDARY LINK
ASSIGN MESSAGE TO PRIMARY LINK
1
C
1
E
ARE THERE ANY LINK DELIVERIES
NO
1
A
YES
ASSEMBLE ALL DELIVERIES ON SAME LINK
4560
GENERATE MESSAGE STATUS
SUBROUTINE MSINQ PLACE MSQ IN QUEUE
IS SIMULATOR IN EDIT MODE
YES
NO
SUBROUTINE XPOD (PROCESS OUTPUT DATA)
IS QUEUE LIMIT FUNCTIONING
NO
YES
DECREMENT QUEUE AVAILABLE COUNT
MORE DELIVERIES
YES
NO
1
A

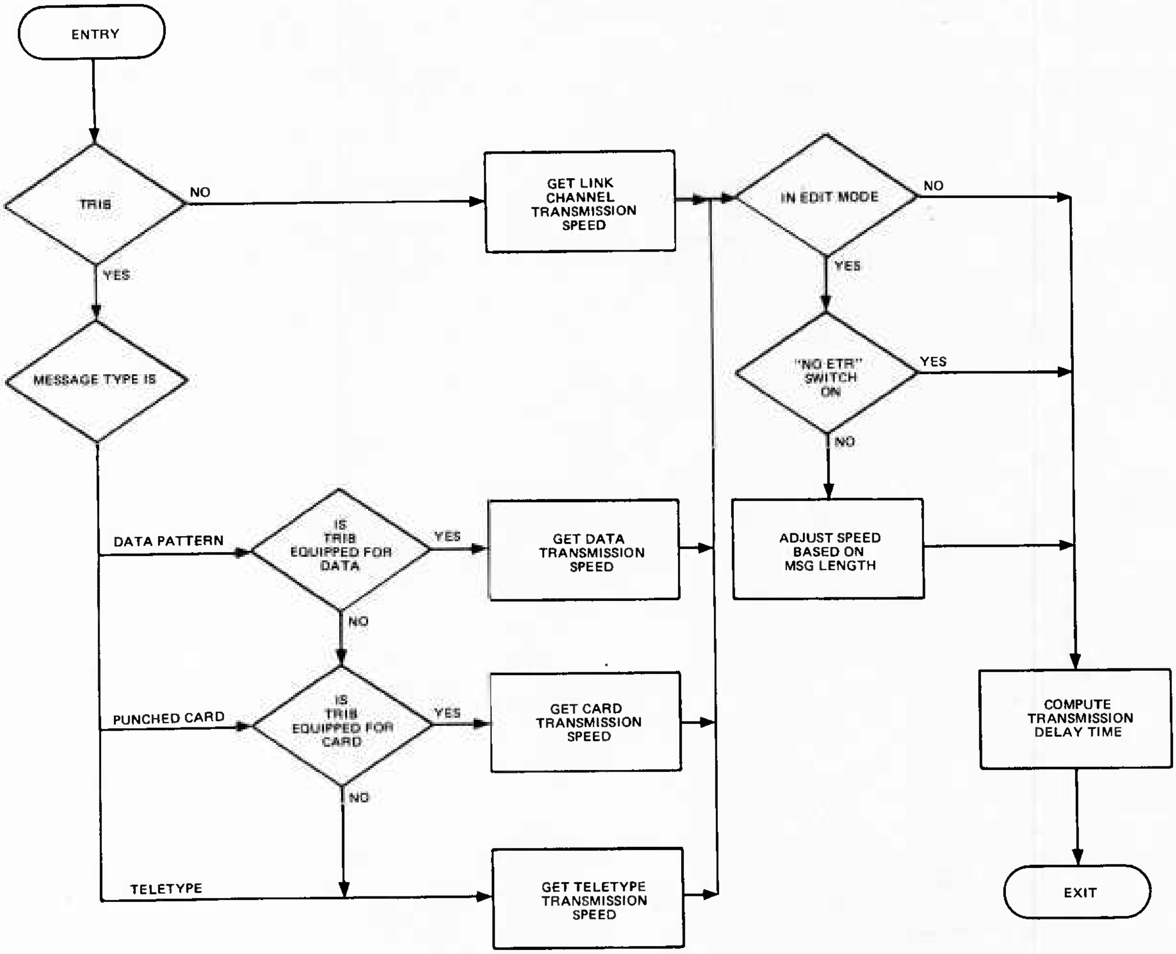
Subroutine GENMS (Sheet 2 of 2)


ENTRY
IS MSG FIRST IN QUEUE
YES
IS MSG ONLY ONE IN QUEUE
YES
CLEAR ALL QUEUE POINTERS
NO
NO
MOVE MSG POINTER TO NEXT IN QUEUE
MOVE NEXT MESSAGE TO QUEUE START POINTER
IS MSG THE LAST PRIORITY MSG
NO
YES
NO
IS THIS THE MESSAGE
YES
IS THIS THE LAST IN QUEUE
YES
MOVE PREVIOUS MESSAGE INTO LAST MESSAGE POINTER
NO
LINK NEXT MESSAGE TO PREVIOUS MESSAGE
IS MSG THE LAST PRIORITY MSG
NO
YES
IS PREVIOUS MSG A PRIORITY
NO
CLEAR LAST PRIORITY MSG POINTER
YES
MOVE PREVIOUS MSG TO LAST PRIORITY MSG POINTER
TURN OFF TRANSMISSION BIT. CLEAR LINKAGE FIELD
EXIT

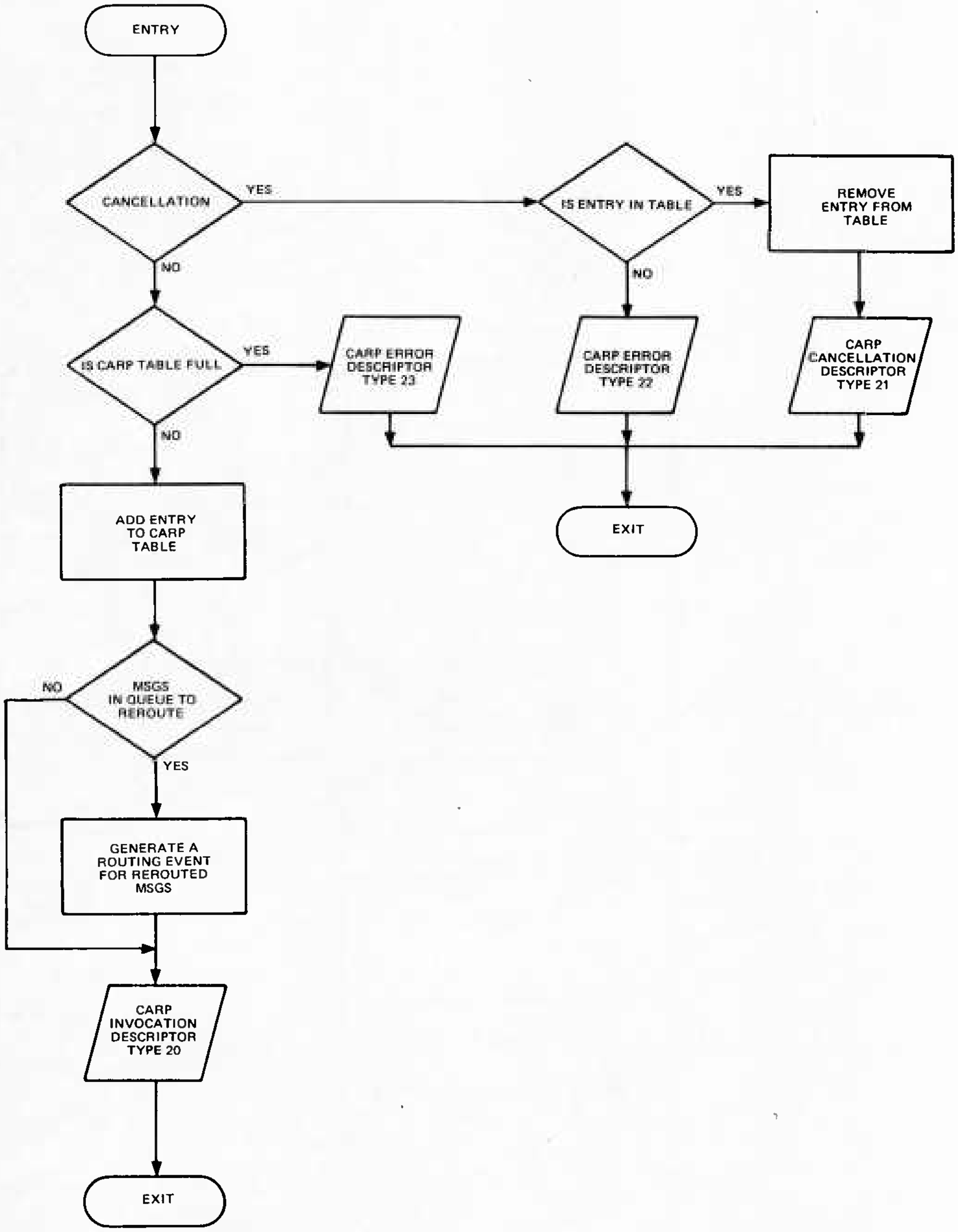
Subroutine UNLNKQ (Sheet 1 of 1)


ENTRY
MESSAGE IN QUEUE DESCRIPTOR TYPE A
IS QUEUE EMPTY
YES
NO
PUT MSG IN FIRST AND LAST MSG POINTERS
IS MSG LOWER PRIORITY THAN LAST
YES
NO
PLACE MSG LAST IN QUEUE
IS MSG HIGHER PRIORITY THAN FIRST
YES
NO
PLACE MSG FIRST IN QUEUE
IS MSG A "PRIORITY" TYPE
YES
NO
IS THERE A "PRIORITY" POINTER
YES
NO
LINK THIS AFTER LAST "PRIORITY" TYPE MSG
MOVE MSG POINTER TO NEXT IN QUEUE
IS MSG LOWER THAN THIS ONE
YES
NO
LINK MESSAGE INTO QUEUE
A
A
IS MSG A "PRIORITY TYPE"
NO
YES
PLACE NEW MSG IN "PRIORITY" POINTER
ADJUST QUEUE COUNTERS
IS QUEUE FLAG SET
YES
NO
IS PRELOAD INDICATOR ON
NO
YES
IS MSG A HIGH PRIORITY
NO
YES
IS LINK OUT OF SERVICE
YES
NO
SET QUEUE FLAG
IN EDIT MODE
YES
NO
ENTER LINK OR TRIB IN PROCESS OUTPUT QUEUE
EXIT

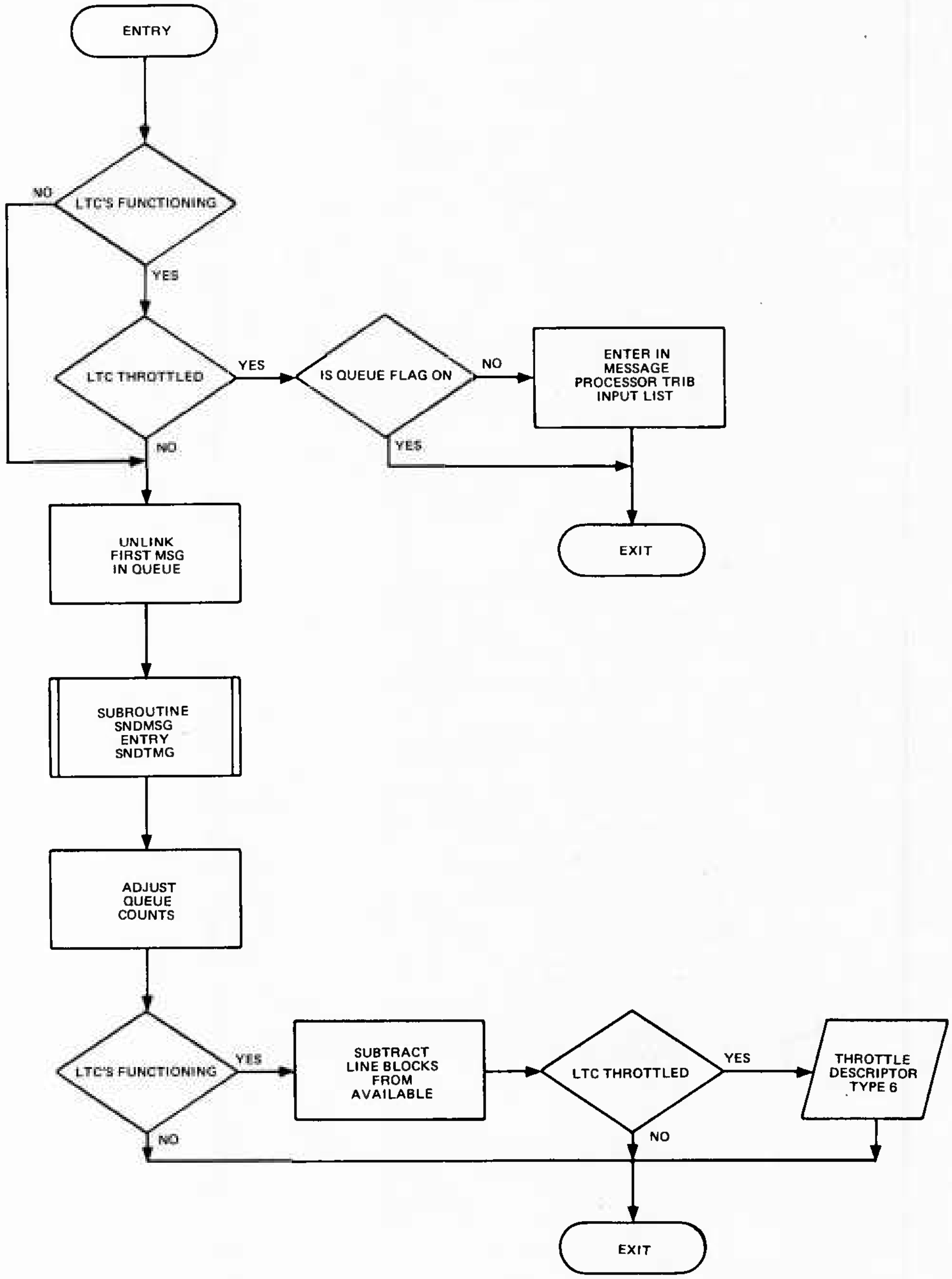
Subroutine MSINQ (Sheet 1 of 1)


ENTRY
TRIB
NO
YES
GET LINK CHANNEL TRANSMISSION SPEED
IN EDIT MODE
NO
YES
"NO ETR" SWITCH ON
YES
NO
MESSAGE TYPE IS
DATA PATTERN
IS TRIB EQUIPPED FOR DATA
YES
NO
GET DATA TRANSMISSION SPEED
ADJUST SPEED BASED ON MSG LENGTH
PUNCHED CARD
IS TRIB EQUIPPED FOR CARD
YES
NO
GET CARD TRANSMISSION SPEED
COMPUTE TRANSMISSION DELAY TIME
TELETYPE
GET TELETYPE TRANSMISSION SPEED
EXIT

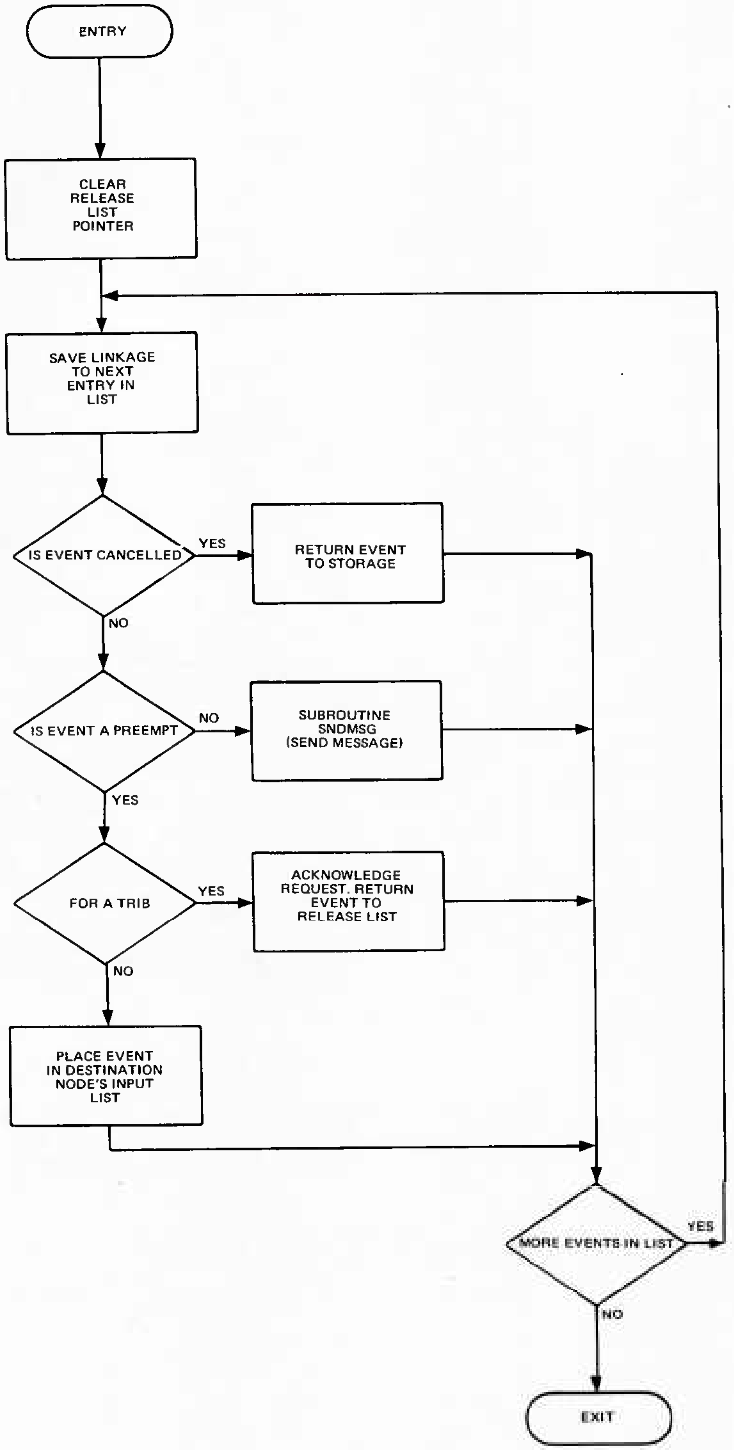
Subroutine XMTIME (Sheet 1 of 1)


ENTRY
CANCELLATION
YES
IS ENTRY IN TABLE
YES
REMOVE ENTRY FROM TABLE
NO
NO
IS CARP TABLE FULL
YES
CARP ERROR DESCRIPTOR TYPE 23
CARP ERROR DESCRIPTOR TYPE 22
CARP CANCELLATION DESCRIPTOR TYPE 21
NO
ADD ENTRY TO CARP TABLE
EXIT
NO
MSGS IN QUEUE TO REROUTE
YES
GENERATE A ROUTING EVENT FOR REROUTED MSGS
CARP INVOCATION DESCRIPTOR TYPE 20
EXIT

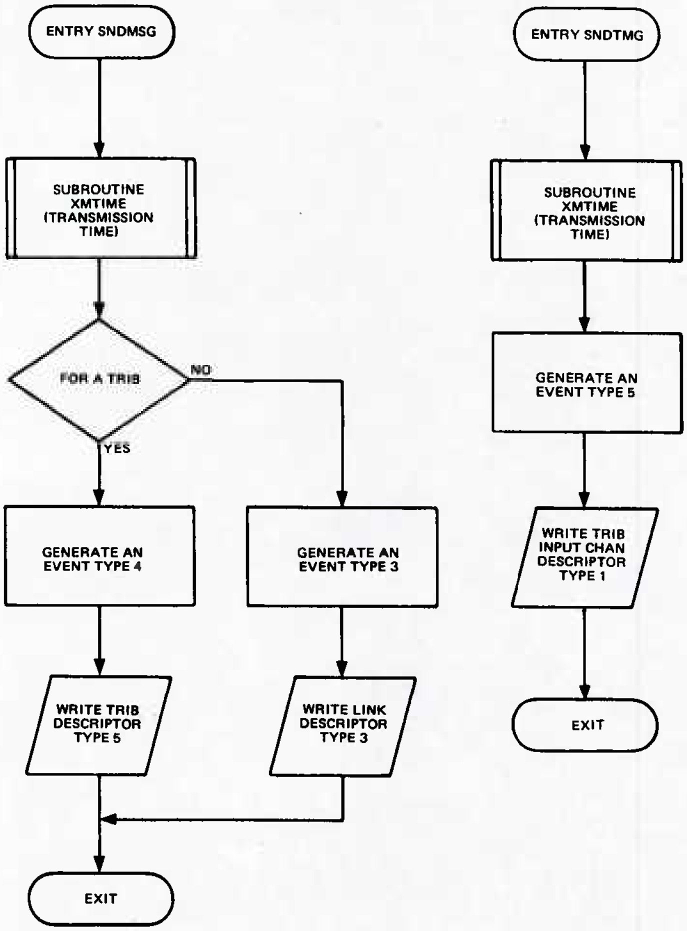
Subroutine CARP (Sheet 1 of 1)


ENTRY
LTC'S FUNCTIONING
NO
YES
LTC THROTTLED
YES
IS QUEUE FLAG ON
NO
ENTER IN MESSAGE PROCESSOR TRIB INPUT LIST
NO
YES
EXIT
UNLINK FIRST MSG IN QUEUE
SUBROUTINE SNDMSG ENTRY SNDTMG
ADJUST QUEUE COUNTS
LTC'S FUNCTIONING
YES
SUBTRACT LINE BLOCKS FROM AVAILABLE
LTC THROTTLED
YES
THROTTLE DESCRIPTOR TYPE 6
NO
NO
EXIT

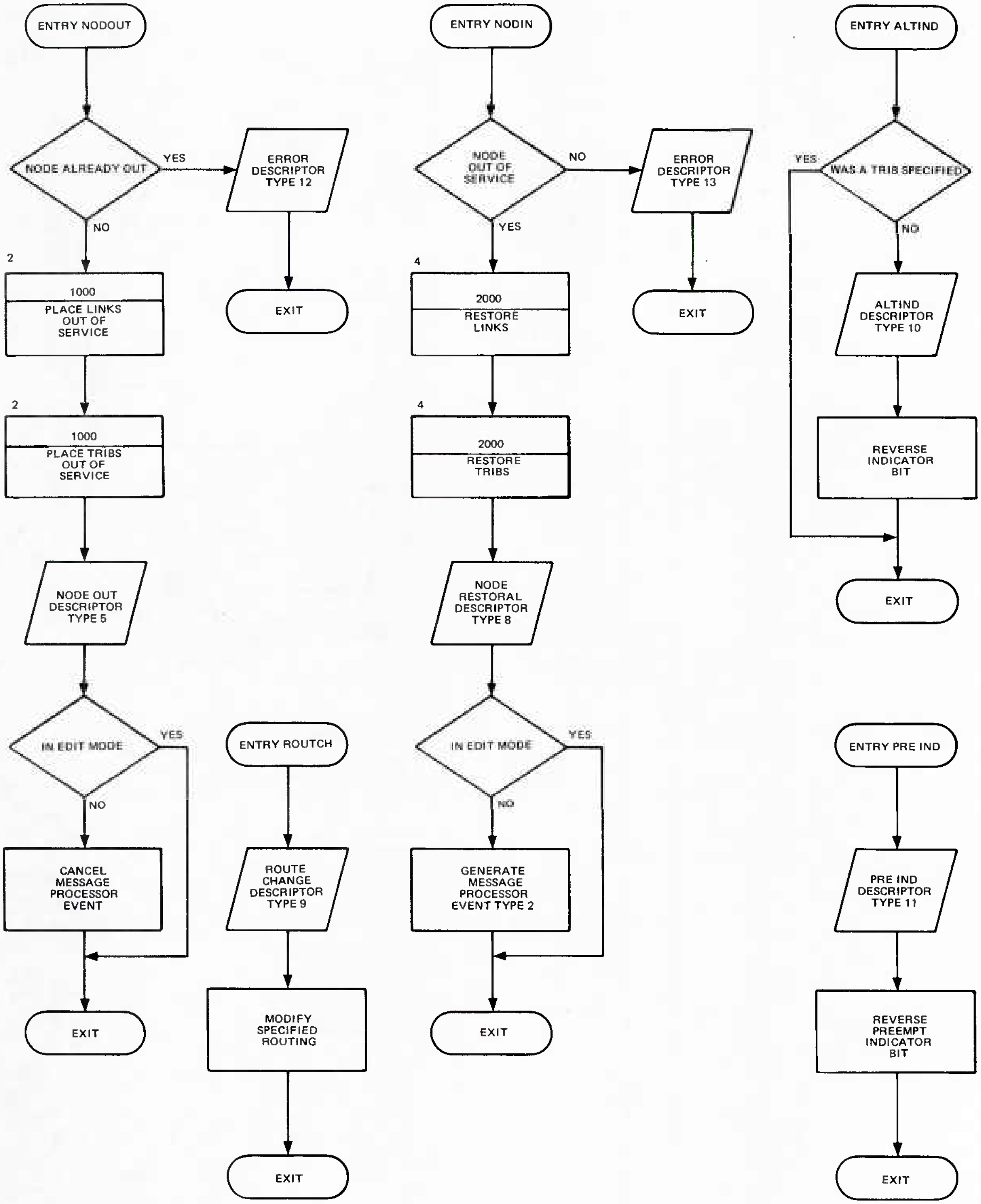
Subroutine NXTMIN (Sheet 1 of 1)


ENTRY
CLEAR RELEASE LIST POINTER
SAVE LINKAGE TO NEXT ENTRY IN LIST
IS EVENT CANCELLED
YES
RETURN EVENT TO STORAGE
NO
IS EVENT A PREEMPT
NO
SUBROUTINE SNDMSG (SEND MESSAGE)
YES
FOR A TRIB
YES
ACKNOWLEDGE REQUEST. RETURN EVENT TO RELEASE LIST
NO
PLACE EVENT IN DESTINATION NODE'S INPUT LIST
MORE EVENTS IN LIST
YES
NO
EXIT

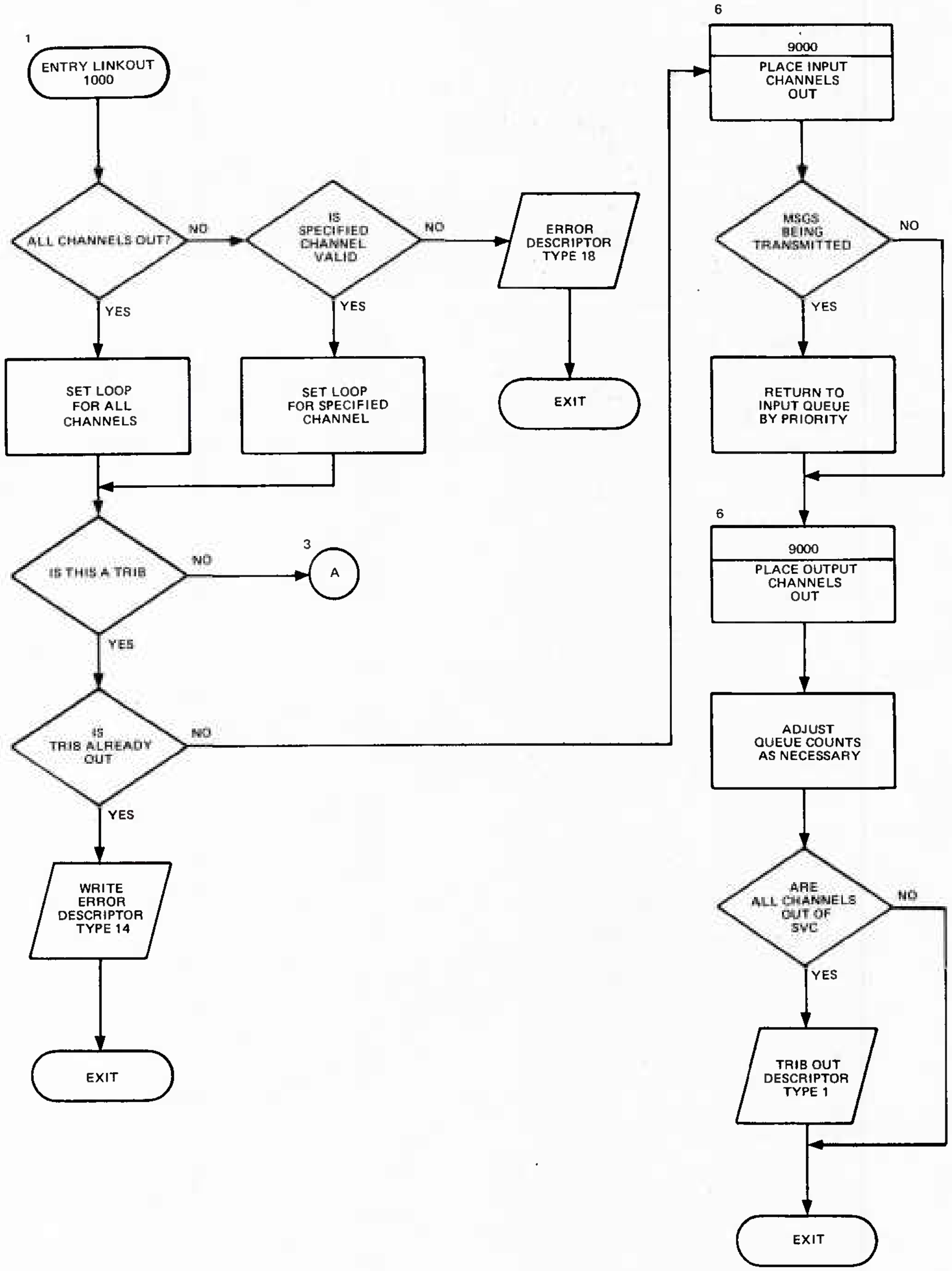
Subroutine RELEAS (Sheet 1 of 1)


ENTRY SNDMSG
SUBROUTINE XMTIME (TRANSMISSION TIME)
FOR A TRIB
NO
YES
GENERATE AN EVENT TYPE 4
GENERATE AN EVENT TYPE 3
WRITE TRIB DESCRIPTOR TYPE 5
WRITE LINK DESCRIPTOR TYPE 3
EXIT
ENTRY SNDTMG
SUBROUTINE XMTIME (TRANSMISSION TIME)
GENERATE AN EVENT TYPE 5
WRITE TRIB INPUT CHAN DESCRIPTOR TYPE 1
EXIT

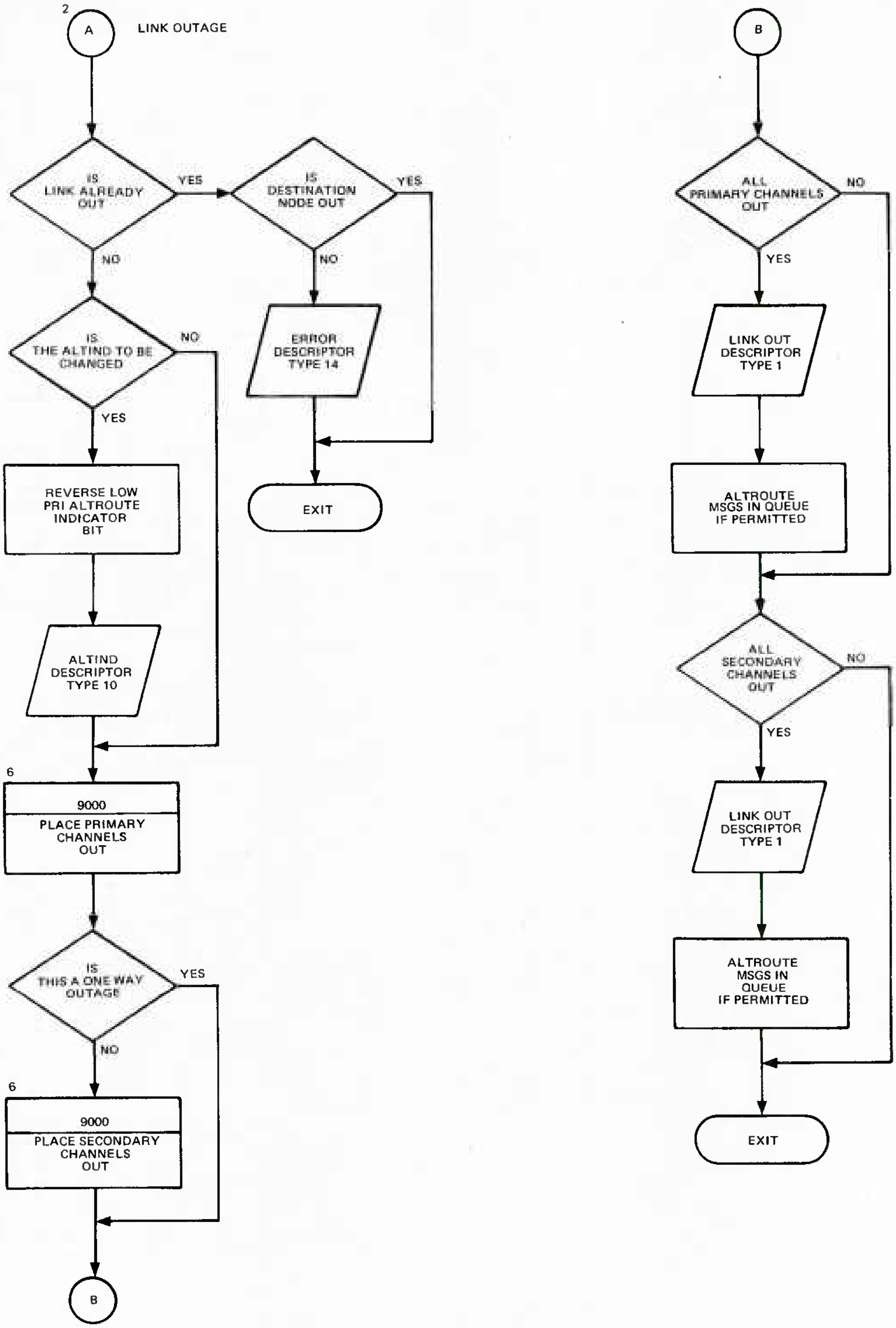
Subroutine SNDMSG (Sheet 1 of 1)



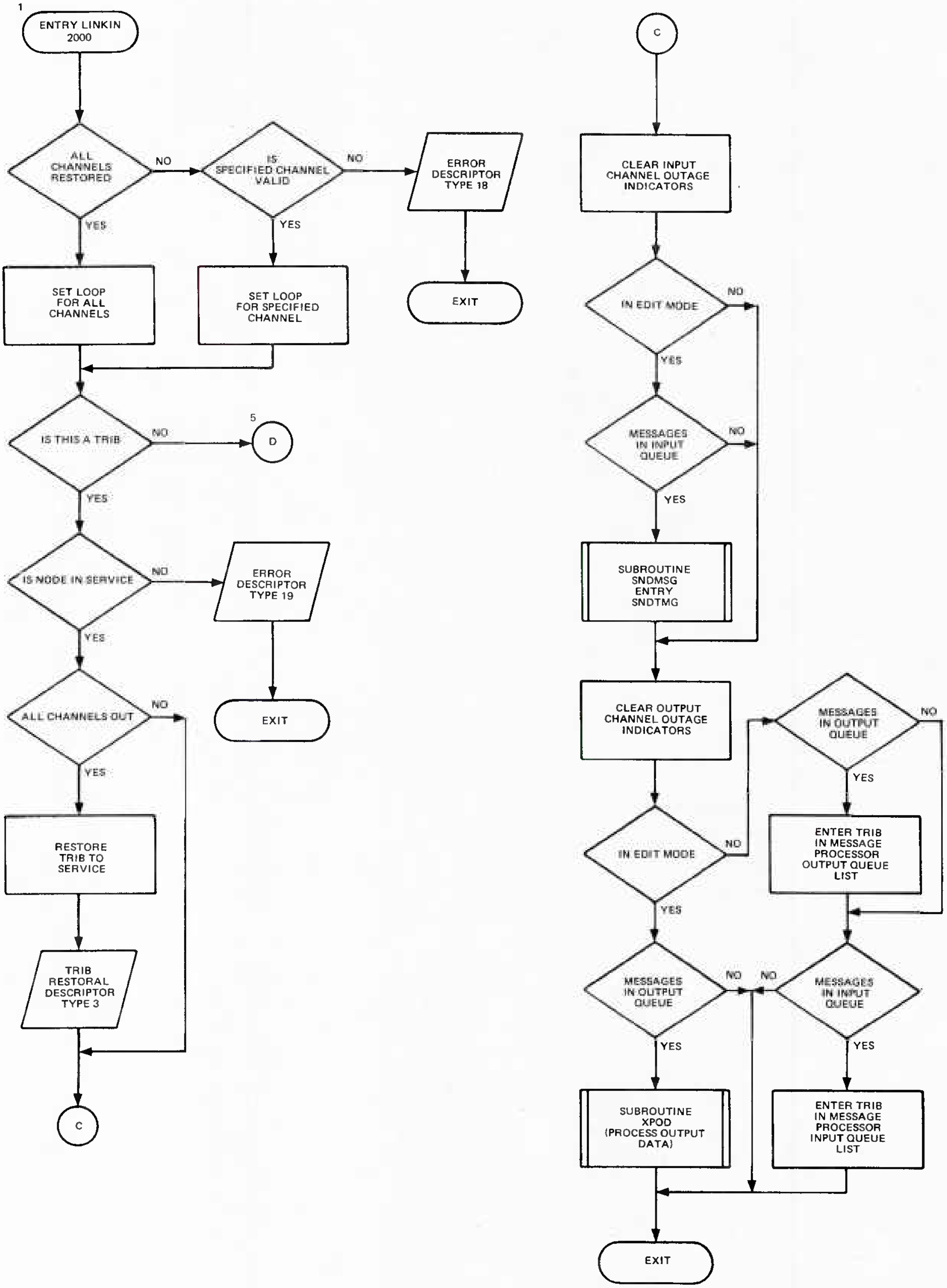
Subroutine SPCVNT (Sheet 1 of 6)


1
ENTRY LINKOUT 1000
ALL CHANNELS OUT?
NO
YES
IS SPECIFIED CHANNEL VALID
NO
YES
ERROR DESCRIPTOR TYPE 18
EXIT
SET LOOP FOR ALL CHANNELS
SET LOOP FOR SPECIFIED CHANNEL
IS THIS A TRIB
NO
3
A
YES
IS TRIB ALREADY OUT
NO
YES
WRITE ERROR DESCRIPTOR TYPE 14
EXIT
6
9000
PLACE INPUT CHANNELS OUT
MSGS BEING TRANSMITTED
NO
YES
RETURN TO INPUT QUEUE BY PRIORITY
6
9000
PLACE OUTPUT CHANNELS OUT
ADJUST QUEUE COUNTS AS NECESSARY
ARE ALL CHANNELS OUT OF SVC
NO
YES
TRIB OUT DESCRIPTOR TYPE 1
EXIT

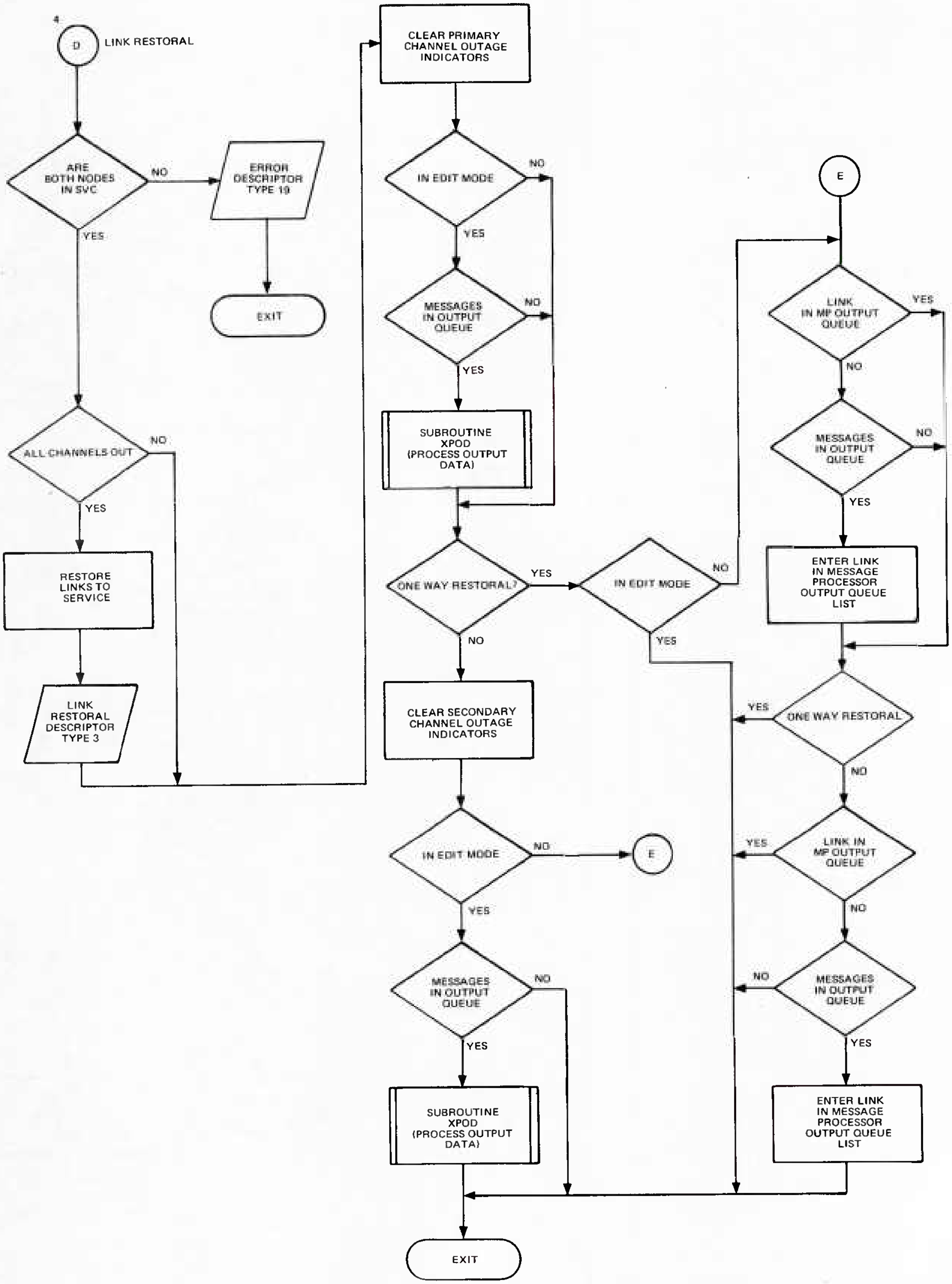
Subroutine SPCVNT (Sheet 2 of 6)


2
A
LINK OUTAGE
IS LINK ALREADY OUT
YES
NO
IS DESTINATION NODE OUT
YES
NO
ERROR DESCRIPTOR TYPE 14
EXIT
IS THE ALTIND TO BE CHANGED
NO
YES
REVERSE LOW PRI ALTROUTE INDICATOR BIT
ALTIND DESCRIPTOR TYPE 10
6
9000
PLACE PRIMARY CHANNELS OUT
IS THIS A ONE WAY OUTAGE
YES
NO
6
9000
PLACE SECONDARY CHANNELS OUT
B
B
ALL PRIMARY CHANNELS OUT
NO
YES
LINK OUT DESCRIPTOR TYPE 1
ALTROUTE MSGS IN QUEUE IF PERMITTED
ALL SECONDARY CHANNELS OUT
NO
YES
LINK OUT DESCRIPTOR TYPE 1
ALTROUTE MSGS IN QUEUE IF PERMITTED
EXIT

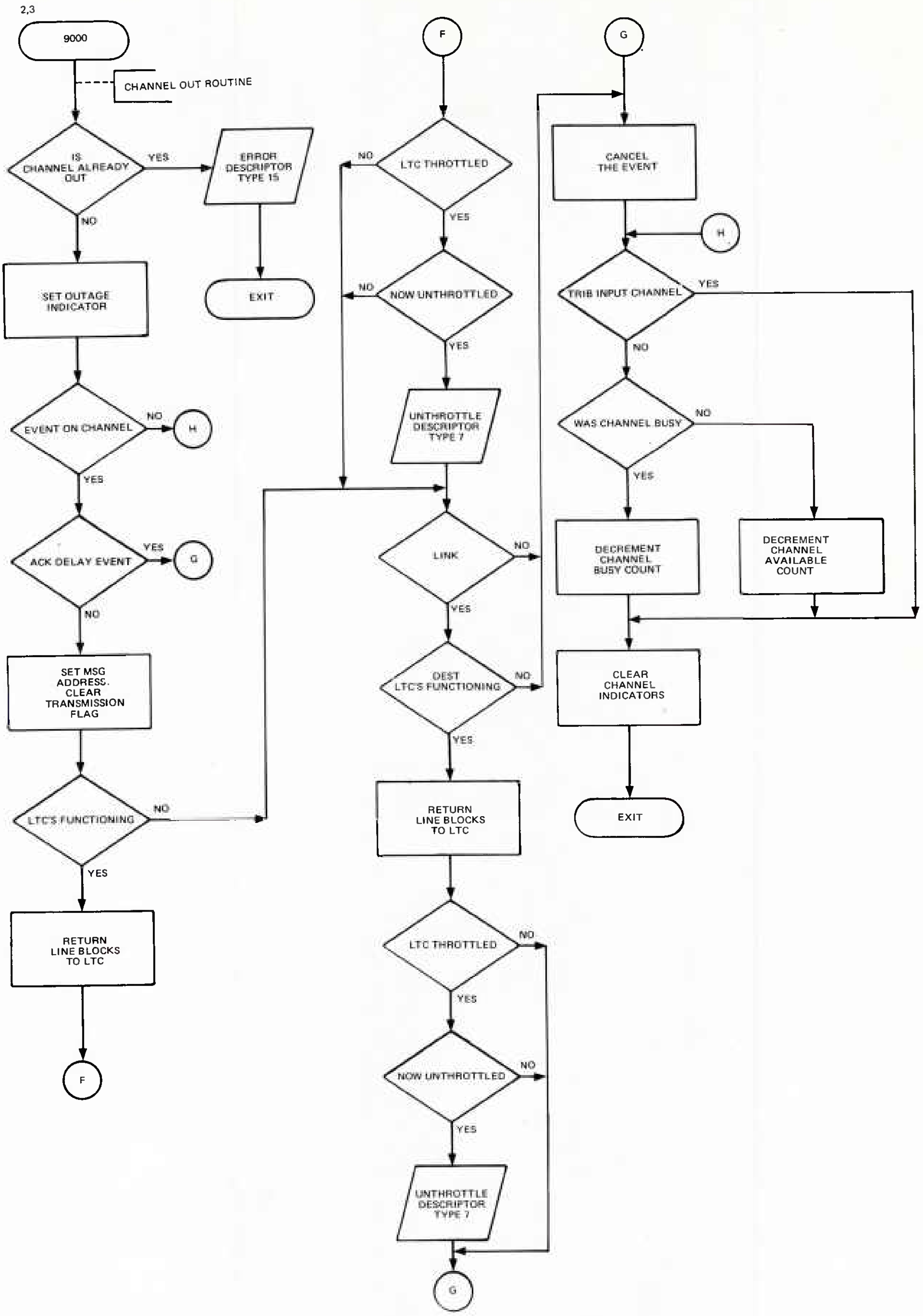
Subroutine SPCVNT (Sheet 3 of 6)


1
ENTRY LINKIN 2000
ALL CHANNELS RESTORED
NO
YES
IS SPECIFIED CHANNEL VALID
NO
YES
ERROR DESCRIPTOR TYPE 18
EXIT
SET LOOP FOR ALL CHANNELS
SET LOOP FOR SPECIFIED CHANNEL
IS THIS A TRIB
NO
5
D
YES
IS NODE IN SERVICE
NO
ERROR DESCRIPTOR TYPE 19
YES
EXIT
ALL CHANNELS OUT
NO
YES
RESTORE TRIB TO SERVICE
TRIB RESTORAL DESCRIPTOR TYPE 3
C
C
CLEAR INPUT CHANNEL OUTAGE INDICATORS
IN EDIT MODE
NO
YES
MESSAGES IN INPUT QUEUE
NO
YES
SUBROUTINE SNDMSG ENTRY SNDTMG
CLEAR OUTPUT CHANNEL OUTAGE INDICATORS
MESSAGES IN OUTPUT QUEUE
NO
YES
ENTER TRIB IN MESSAGE PROCESSOR OUTPUT QUEUE LIST
IN EDIT MODE
NO
YES
MESSAGES IN OUTPUT QUEUE
NO
NO
MESSAGES IN INPUT QUEUE
YES
YES
SUBROUTINE XPOD (PROCESS OUTPUT DATA)
ENTER TRIB IN MESSAGE PROCESSOR INPUT QUEUE LIST
EXIT

Subroutine SPCVNT (Sheet 4 of 6)



Subroutine SPCVNT (Sheet 5 of 6)



Subroutine SPCVNT (Sheet 6 of 6)

SECTION 7 - REPORTS PROGRAM

7.1 PROGRAM REQUIREMENTS DESCRIPTION

7.1.1 Program Name: Reports Program

7.1.2 Program Identification: DNREPTS1

7.1.3 Purpose

The Reports program summarizes the results of the simulation performed by the preceding programs. The event descriptor records written by the Basic Simulation program are used to provide Speed of Service (SOS), Network Delay, Link Utilization, and In-Station Handling Time Reports. The status records are used to provide Switch Queue, Link Queue, and Tributary Queue Reports.

7.1.4 Requirements

The Reports program source language is ANSI COBOL which is compatible with the IBM 360/OS environment.

Core requirements have been minimized by the segmentation feature available in the COBOL language. The program requires approximately 200K bytes of memory on an IBM 360 System for execution.

7.2 INPUT SPECIFICATIONS

The Reports program requires five separate data sets when all options are exercised. One of the five, Option Run cards, is required regardless of the options desired.

7.2.1 Translation Tables

The program uses the Translation file produced by the Status Generator program to cross-reference the component identification used by the Basic Simulation program with the component names used in the model. This file is also used to obtain data on certain components which are necessary to program computations.

The Translation Tables are generated by the Status Generator Program and are described in Paragraph 2.3.3.

7.2.2 Events Files

When the SOS Report is requested, the SOS Event file, generated by the Basic Simulation program, is a required input. When Network Delay, Link Utilization, or In-Station Handling Time Reports are requested, the Link Event file is a required input.

The SOS Event file is generated by the Basic Simulation program and is described in Paragraph 6.3.1. The Link Event file is generated by the Basic Simulation program and is described in Paragraph 6.3.2.

7.2.3 Status File

When the reports requested include Switch Queue, Link Queue, or Tributary Queue Reports, the Status file generated by the Basic Simulation program is a required input and is described in Paragraph 6.3.3.

7.2.4 Option Run Cards

Option Run cards are used to control the format and content of output listings. Coding conventions for these cards are described in Tables 7-1 through 7-11. The applicability of each card to the various reports is shown in Table 7-12.

7.2.4.1 Type Report Cards

These cards control the output report format and also determine other data sets that are required for successful program execution. Coding conventions are rigorous in that character-by-character scanning of these cards is not performed, i.e., a field test is made on the first 12 positions for each allowable report name via COBOL condition-names (level 88). The allowable entries on Type Report cards are:

SOS REPORT
IN-STATION
SWITCH QUEUE
NET DELAY
LINK UTILITY
LINK QUEUE
TRIB QUEUE

7.2.4.2 Time Cards

Time card options allow the selection of a particular time frame to observe Simulation activities. If it is desired that these reports cover some time frame less than the period of simulation, start and stop times may be specified.

Only two forms of the Time card are permissible:

```
START, HHMM

STOP, HHMM
```

where HHMM is the hour and minute in military form. If no Time cards are submitted, the total simulator output is processed.

7.2.4.3 LMF Card

The Language Media Format (LMF) card permits selection of a single LMF for observation. A condition-name field test on positions 1-4 for LMF= is used to determine the presence of this card. The single numeric character following the LMF= is stored for comparison against input records. Any record comparison matching the stored value is passed for further processing. Mismatches are bypassed. In the absence of an LMF card, no comparisons are made and all records are processed.

7.2.4.4 Queue Sample Index Card

The Queue Sample Index value is a means of observing Link or Switch Queue status on a fixed, selective basis. A condition-name test on the first five positions for TIME= is used to determine the presence of this option. The number following this entry is stored for comparison against a tally of records passed to the queue print routines. This index (N) is some value between 1 and 9999 which, in effect, tells the program to print the first record, skip N records, print the next record, et cetera. The Queue Sample Index is disregarded in determining the maximum queue size data at the end of each report. In the absence of a Queue Sample Index, all status records are allowed to print.

7.2.4.5 Interval Cards

An Interval card permits observation of SOS, Network Delay, and In Station Handling times at class intervals other than those initialized in the program. Initial upper-class limits are 60, 120, 180, 300, 900, 1200, 1800, 2400, 3000, 3600, 5400, 7200, 10800, 14400, and 99999. A condition-name test on the first 11 positions for INTERVALS, Δ is made to determine the presence of this option. Up to 17 upper-class limit values may be entered through this option, possibly necessitating a subsequent card. When a second card is required, it must immediately follow the Interval card or unpredictable program results occur. Coding on the second card may start in any column.

Once the intervals have been altered they remain at the new values unless altered by a subsequent Interval card in a new report request. Jobs requesting more than one report must consider this factor and care should be exercised in sequencing multiple requests.

7.2.4.6 All Tribs Cards

This option is applicable only if the SOS Report option was present on a Type Run card. Precedence of the All Tribs option is determined by a condition-name test on the first 12 positions for ALLΔTRIBS ΔΔΔ.

When present, control switches are set in the SOS routines to ignore all but source and destination tributary traffic.

7.2.4.7 Trib Pair Cards

This option is applicable only when the SOS Report is being processed. It permits selective observation of traffic between specific source and destination tributaries. Presence of the option is determined by a condition-name test for TRIBΔ(in the first six positions. The source and destination tributary names then follow. Each pair of source and destination tributary names is placed in a table and controls are established to allow passage of only those records with matching source and destination combinations.

7.2.4.8 All Switches Card

This option is applicable only if the SOS Report was requested. A condition-name test is made for ALL SWITCHES on the first 12 positions. This option establishes the controls to pass only source and destination traffic between switches. An additional control is established to summarize the total message count and traffic time between each pair of switches for production of the Average SOS Report.

7.2.4.9 Switch Pair Cards

This option is applicable only if the SOS Report was requested. A condition-name test is made for SWITCHΔ(in the first eight positions to determine the presence of this option. The source and destination switch names that follow are placed in a table and controls are established to allow processing of only those records with matching source and destination switches. The Average SOS controls are also established.

7.2.4.10 Link Cards

This option is applicable only if the Link Queue Report is requested and its presence is determined by a condition-name test for LINKΔ in the first five positions. Each link name is placed in a table and only those records with matching link names are allowed to process. In the absence of Link cards, all links encountered are processed.

7.2.4.11 Switch Cards

Applicable only in conjunction with the Switch Queue Report, this option is detected by a condition-name test for SWITCH in the first six positions. Each switch name is placed in a table and only those records with a matching switch name are allowed to process. All switches are processed if this option is not present.

7.2.4.12 End of Request

Operation of the Reports program permits more than one report to be requested in a single program execution. The end of each individual report request and its associated Option Run cards is detected by a condition-name test for END DATA in the first eight positions. When this option is encountered, program control passes to the processing routines for the requested report. At the end of the report, processing control returns to process the next report request.

7.3 OUTPUT SPECIFICATION

The Reports program produces up to seven different reports, depending on the options selected. In addition, the Option Run cards submitted for each report are listed immediately prior to the report they control.

7.3.1 Speed of Service Reports

The SOS in a network is determined by computing the elapsed time from when a message starts into the network at the originating tributary until delivery is complete at the destination tributary. When this report is requested, the program also produces a system (all messages in the complete network) report where the average SOS for each precedence level is displayed. To assist in analysis, a matrix display is presented where the x axis is divided into 17 time intervals and the y axis into the five precedence levels. At the intersections are counts of the number of messages of that precedence which were delivered within that time interval. A second matrix of the same type is displayed where the counts are cumulative. In tabular format the mean SOS, standard deviation, minimum SOS and message number, and maximum SOS and message number are displayed for each precedence level. The simulation period that the report covers is displayed in entries titled Start and Stop.

As a minimum, the system report is produced whenever the SOS Report is requested. The various options for this report allow separate reports to be requested showing the SOS between each of the network switch pairs, between specified switch pairs, between each of the network tributary pairs, or between specified tributary pairs. Only one of these options may be specified in a single program execution and, to prevent waste of computer time and materials, the program will not exercise the SOS between each of the network tributary pairs if there are more than 50 tributaries. If it is desired that these reports cover some time frame less than the period of simulation, the start and stop times may be specified. Normally, the report covers all messages regardless of their language media format; however, one type of format may be selected to be covered. Still another option is the ability to change the time intervals used in the matrix portion of the report.

Whenever the All Switch Pairs or Specified Switch Pairs option is selected an average SOS between designated switches will be produced. This report is in a matrix format.

7.3.2 Network Delay Report

The Network Delay Report is similar to the SOS Report in concept and options. The difference between the reports is in the computation of elapsed time. This report demonstrates the efficiency of the switching centers and the interswitch links in the network. The elapsed time computation for each message excludes the time of transmission between the tributaries and switches and the time spent in outgoing tributary queues. The elapsed time does include the in-station handling time at the switches, the interswitch transmission time, and the time spent in interswitch link queues.

7.3.3 In-Station Handling Time Report

The in-station handling time of a message is a computation of the elapsed time from when an incoming message to a switch is first available for processing until that message begins outward transmission on an interswitch link or is placed in an outgoing tributary queue. When the In-Station Handling Time Report is requested, the program prepares a report for each of the network switches. The format of this report is similar to that used for the SOS Report.

The options available to this report are start and stop time specifications, Language Media Format specification, and modification of the time intervals on the matrix portion of the report.

7.3.4 Link Utilization Report

The Link Utilization Report summarizes the effective use of an interswitch link. When this report is requested, the program prepares a report on each direction of each interswitch link. The report includes static link characteristics such as model name, source switch, destination switch, and transmission speed, and summarizes, by precedence, the number of completed messages carried (excluding partial messages which were preempted) and the number of line blocks transmitted (including partial messages). Also indicated in the average length of completed messages. The link capacity is computed and displayed. This computation is derived from the speed of the link and the time it was available (excluding outages). The link utilization factor (a percentage based upon the time the link was in use and the time it was available) is computed and displayed.

When a link is composed of two or more channels of different transmission speeds, a separate report is prepared for each speed group.

The options available for this report are start and stop time specification and Language Media Format specification.

7.3.5 Switch Queue Report

When the Switch Status Delay parameter is specified in the Basic Simulation program, a status record for each switch is written at the specified intervals. The record contains the count of messages in tributary input queues, tributary output queues, and in link queues. When the Switch Queue Report is requested, these records are sorted and written in tabular format. At the conclusion of each report, a notation of the largest queue size for each of the three queues is made with the time that the maximum occurred.

The switches for reports desired may be specified. If none is specified, the report is made for all switches. The start and stop times may be specified. To reduce the listing for an initial report, a Queue Sample Index may be specified. This index (N) is of some value between 1 and 9999 which, in effect, tells the program to print the first record, skip N records, print the next record, et cetera. The largest queue notation referenced in the preceding paragraph disregards the Queue Sample Index, but is restricted by the start and stop time.

7.3.6 Link Queue Report

The Link Queue Report is similar to the Switch Queue Report, except that the queues shown are only for links and are divided into high-precedence and low-precedence counts. The options are identical, except that desired links may be specified.

7.3.7 Tributary Queue Report

The Tributary Status records are written in the same manner as the Switch and Link Status records; however, the number of tributaries which normally exist in the AUTODIN network precludes the printing of a report similar to the Switch or Link Queue Reports. The report is tabular and contains one entry for each tributary in the network indicating the maximum queue which existed and the time of occurrence. If the queue size for any tributary warrants further information, it is obtained using the List Events program.

The only options available to this report are the start and stop time specifications.

7.3.8 Option Run Card Listing

Each set of Option Run cards submitted is listed immediately prior to the report it controls. The spaces indicated on the Option Run card forms may be utilized for comments which will appear on the listing.

7.4 PROGRAM LOGIC

The Initial or Root Segment consists of two distinct phases:

Phase 1. Reading, printing, and interpreting the Option Run cards to establish program control values and report options.

Phase 2. The base routines that call the overlay segments to perform the actual report processing.

Program operation begins and terminates with this segment and provides the return mechanism for multiple request jobs. For these reasons, it has been assigned a segment priority number less than the segment limit and is permanently resident in core. With the exception of several other minor subroutines common to several overlays, all segments are dynamic.

7.4.1 Phase 1

1. Read Run Card - At program initiation, each Option Run card is read, immediately printed, and passed to a series of tests to determine its character. These tests are based on condition-name entries (level 88) established in the FD area of the RUN-CARD file record descriptions.

2. Report Type Tests - Testing is performed for each allowable type of report. A value corresponding to each report name is set with appropriate condition-names for each report type. Later portions of the program utilize these condition-names to establish the control paths used to produce the report specified.

3. Start and Stop Time Tests - If either of these options are encountered, conversion of military time format into seconds is performed and the latter stored to establish the start and stop points of the observation period. The military time format is also stored for use on output listings. A condition-name value is set to indicate that these options were encountered.

4. Switch and Link, Card Switch and Trib Pair Tests - When these options are encountered, appropriate condition-name values are set indicating their presence. Tallies are kept on the number of entries as each user designation is placed in its appropriate table. These tables are subsequently used for screening in conjunction with the type of report selected for printing.

5. LMF Test - If an LMF entry is encountered, the value is stored for screening in subsequent routines.

6. All Switches and Tribs - When these options are encountered, condition-name values are set to indicate their presence.

7. Interval Modification - Interval modification is performed by overlaying the initialized table of upper-class limits. Examination of the Interval cards is by character scanning. Each character encountered on the Interval card is moved to an area for assembly as an upper-class limit value.

When the comma (,) delimitor between values is encountered, the assembled value overlays the original table entry. Imbedded blanks are ignored by the character scan and detection of the ($) character causes an exit from the routine.

8. Queue Sample Index - If present, the Queue Sample Index is moved to a condition-name field and used for screening against queue records.

9. End of Report Request - If an End Data card is encountered, control is transferred to the exit routines for the Phase-1 section.

10. Error Card Trap - Any card in the report request stream not meeting the preceding condition-name tests is ignored and control returned is to the card read routine.

11. Exit Routines - After the request stream has been edited, a series of tests are performed to evaluate the validity and correspondence of the options selected with program requirements. The absence of a Type Record card is critical. If this situation occurs, an appropriate message is generated and program termination occurs. If conflicting parameters for a particular report have been entered, a message conveying this is printed and program operation continues in accordance with preestablished defaults.

A final routine is invoked to reformat upper-class limit tables for printing, if the intervals were modified. Control then passes to the base processing routines.

7.4.2 Phase 2

As described earlier, a condition-name value commensurate with each report was established from the Type Report card. Phase 2 begins with a series of tests for these condition-names with transfers of control to the appropriate base routines for the selected report.

Because there is a high degree of commonality in the initial stages of their processing, the SOS, Link Utilization, Network Delay, and In-Station Handling Time Reports are grouped for discussion purposes. Departure from this grouping will occur when processing for a report becomes unique. A similar approach will be used to discuss the Switch, Link, and Trib Queue Reports.

For convenience the former set of reports will be referred to as Group 1 and the Queue reports as Group 2.

7.4.2.1 Group 1 Reports

Initially, the Translation Tables are read into memory (LOAD-XLATE-TABLES Section) to serve as reference authorities for all but the Link Utilization Report. Two tables are utilized for this purpose. As each Translation Table record is read and determined to be either a Tributary or Switch record, the model name and assigned numeric identifier are moved to their respective trib or switch tables under control of the same subscript. This establishes a direct relationship between the name and number which is used in later routines to translate input data. Control is then transferred to a COBOL SORT. While the particular sort sequences required for each Group 1 report is different, the associated input procedure feature is ideal for combining a number of common routines related to options established in Phase 1.

Prior to release to the SORT feature, the following functions are performed in the input procedure for Group 1 reports.

1. Input files for the selected report are opened and read until a first clock record (ORAC = Ø) occurs.

2. If there is no Start Time card, the time in seconds on the first clock is stored and converted to HHMM format for printing on output report headings. If there is a Start Time card, the stored value is compared against the record's clock time. If the record's clock time is less than the start time, a scanning loop is entered to read the input file until another clock record is encountered and the test is performed again. If the record's clock time is not less than the start time, the value is stored, a switch is set to bypass the start time test on subsequent clock records, and control is passed to a stop time test.

3. If a Stop Time card is present and the record's clock time is greater, subsequent event record times are set to the stop time and control is passed

to the remaining processing routines. If there is no Stop Time card or the clock time is less than the stop time, the value is stored and another input record read.

4. If the new input record is an event record, several tests are performed. If an LMF card is present, the record is examined for a corresponding LMF (ORTP) value. Records with matching LMF values are passed to subsequent tests while mismatches are bypassed by return to the input read paragraph. If no LMF is specified, this test is bypassed.

5. If a Network Delay report is requested, tests are made on Action Codes and ORAC values converted from 2s to 1s and As to 6s prior to release to SORT.

6. Condition-name tests are preformed to assure processing of only those input records commensurate with the report type specified.

Any record failing these tests is bypassed by return to the input read paragraph. Those meeting the tests are released to the SORT along with the stored clock record time indicating the time of their occurrence.

After processing all input records, a final test is made as to whether a Stop Time card was submitted. If not, the last stored clock record value is kept and converted to HHMM format for printing, appropriate files are closed, and control transferred to the SORT.

Upon completion of the SORT and return to the base routines, processing of the Group 1 reports begins to diverge. Each separate report path will be described.

7.4.2.1.1 Speed of Service and Network Delay

To this point, two major steps have occurred in the processing of these reports:

1. Screening for the proper observation period, LMF, and record types.

2. Sequencing the file by message number (ORNUM), record type (ORAC), and time of occurrence (ORT).

Before the data is ready for printing, several additional steps must be performed. Additional screening for Tributary or Switch Pair options, translation of switch or tributary numbers to names, and resequencing the file into report sequence are performed with the next dynamic segment. These steps, and printing, are accomplished by the COBOL SORT feature and associated input and output procedures.

The output file from the first SORT, FIRST-WORK-FILE, is opened and read by the SORT input procedure, SECOND-SCREEN-SORT Section. If it is a message-entry-into-the-system (ORAC=1), the message number and time are stored. To avoid unnecessary processing where the System Summary only will be produced, a bypass has been established. This option (System Summary only) is tested for and, if present, source and destination fields in the output area are cleared and the next record is read. This is permissible since only one source and destination are required for the System Summary and that need only be a constant for all records. If the All Switches or Switch Pair options are present, the switch number (ORSW) is translated into its user name by a lookup in the previously loaded Translation Tables. If the All Switches option is present, the translated switch name is moved to the output area and control returns to the input read paragraph. Had the Switch Pairs option been specified, an additional lookup is performed on the translated switch name to determine if it was one of the source switches specified. If so, the translated switch name is moved to the output area and control returns to the input read paragraph. If not, the record is bypassed and another record read. Tributary options are processed in a like manner.

If the next record is a delivery, translation and/or screening are performed on the destination switch or tributary as was done with message entries. After the translated destination name, message number, and precedence are moved to the output area, the record is released to the final SORT. This process continues until all records on the input file have been read and control then passes to the SORT. The SORT returns the edited and screened message data to the final step (output procedure) sequenced by destination, origin, precedence, and time.

The steps that must now be performed are to tally the number of messages transmitted between each source and destination by their time-in-system and level of precedence. This data must then be formatted and printed.

To do this, the following process is performed by a SORT output procedure, PRINT-SOS-ISH-REPORT Section.

1. Tally Routine - As each record is returned from the final SORT a test is performed to determine whether the source/destination combination has changed. A change indicates the end of processing for this combination and control transfers to the Print routines. If the combination is unchanged, a loop is entered to compare the message time-in-system to the upper-class limits of the Interval Tables.

   When the appropriate class interval is located, a tally of 1 is made to a class-frequency matrix dimensioned by precedence and interval; the message duration time added to a table dimensioned by precedence; and a sum-of-squares table augmented by the time-in-system squared. In addition, areas used to determine average message time and standard deviations at each level of precedence are augmented. A comparison of the message time-in-system is made against tables of minimum and maximum times to determine those values for each record at each level of precedence. Control then returns to the input read paragraph.

2. Print Routines - For each combination a one-page listing is generated that displays a matrix of the tallied information by level of precedence and message time-in-system. This display is formatted under control of PERFORM loops that first sum the matrix by rows (levels of precedence) and columns (class intervals), secondly, edit and format each line to be printed, and thirdly, print each line. This process is performed until all levels of precedence and a total line have been printed. The initial matrix is then added to a system class frequency matrix for producing the System Summary Report.

Another matrix listed for each source/destination combination contains the cumulative class frequencies of the initial matrix. This matrix is also produced under control of PERFORM loops, formatted, and printed in a similar manner.

Finally, a summary of average and standard deviation times for each level of precedence, minimum and maximum message times, and their associated message numbers are listed. Formulas for the average and standard deviation times at each level or are:

$$\text{Mean-Message-Time} = \frac{\text{Cumulative Message Time}}{\sum \text{Class frequency of each interval}}$$

$$\text{Standard Deviation} = \sqrt{\frac{\sum \text{ (Message Times}^2\text{) - (Mean-Message Time)}^2}{\sum \text{Class frequency of each interval}}}$$

These times are also computed for all levels of precedence to produce a total for the source/destination combination.

All times are converted from seconds to a print format (HH:MM:SS) and listed with the mean, minimum, and maximum time data at each level of precedence and in total. Cumulative and minimum/maximum message data is moved to the System Summary work area, class frequency matrix work areas are re-initialized, and control is returned to process the next source/destination combination.

After all combinations have been processed, System Summary areas are moved to the class frequency matrix areas, switches set to alter header data and the exit to the report process, and a single pass made through the Print routines to produce the System Summary Report. Upon completion of the System Summary Report, control is returned to the Root Segment via the base routines to process additional report requests or to terminate program execution.

7.4.2.1.2 In-Station Handling

The initial SORT for the In-Station Report sequences the FIRST-WORK-FILE by Switch (ORSW), message number (ORNUM), and time of occurrence (ORT). This file is now ready for input to the final process and, because of the similarity to the SOS Report, an appendage is added to the PRINT-SOS-ISH-REPORT Section to accommodate the requirements.

Processing subsequent to the initial SORT begins with alteration of the header lines used by the SOS Reports to be compatible with the In-Station Handling Time Report (ISH).

As each record is returned to the ISH routines the switch number is translated into its designated name, and all input record data is moved to a work area shared by the SOS and Network Delay routines.

As each record is moved into this work area, a test is performed to determine if the switch name has changed. If it has, control passes to the Print routines and processing continues. If the switch has not changed, a series of tests is performed to determine the event type and accumulate the message data for printing. Those tests are as follows:

If the work area record type is an ISH entry (ORAC = 2 or 4), the record time is established as the ENTRY-TIME at the subject switch. If the record type is an exit from a switch (ORAC = 3 or A), the record time is taken as the EXIT-TIME, the difference between the ENTRY-TIME and EXIT-TIME is determined, and control is passed to the SOS routines that tally class frequencies and message time-in-system (in this case, within an individual switch). Appropriate conditional branches are present in the tally routines to assure return of the ISH appendage. As each new message is encountered, this process continues until a change in the switch number occurs.

Processing continues in this manner until all records have been processed, then appropriate switches are set to produce a System Summary and return control to the Root Segment via the base routines. There are no options to select specific switches for processing in the ISH Report.

7.4.2.1.3 Link Utilization Report

The processing required to produce the Link Utilization Report is performed by a COBOL SORT and concomitant input and output procedures. The input procedure contains the data manipulation routines to create the print file for the Link Utilization Report and the output procedure contains the formatting and print routines.

Prior to printing, the number of line-blocks, transmission time, and number of messages transmitted over a particular link must be determined to ascertain the utilization factor. Utilization, as a percentage factor, is determined by comparing the actual utilization of the link in terms of line-blocks transmitted to the link's capacity in terms of potential line transmissions. Determination of this factor is complicated since interruptions occur before final completion of a message transmission and multi-speed channel groups exist on some links. Once these factors have been determined, only editing, formatting, and printing are required to produce the Link Utilization Report.

To this point, a file has been created with link event data (ORAC = 3, 4, 7, 8, or 9) sorted in link (ORNUM), channel (ORCH), message number (ORNUM), and time (ORT) sequence. Now the information necessary to determine the utilization factor is developed. This is accomplished in a SORT input procedure named SECOND-LINK-FILE Section.

The TRANSLATE-TABLES file is opened and positioned at the first link channel record. Because the TRANSLATE-TABLES file treats each direction of the link as a unique entity and because chanels of varying speed within these entities have individual record entries, a comparison is made with the previously stored link number to determine if this is the first record occurrence of this particular link. If so, the number of channels of this speed are moved to a work area. If not, the number of channels is added to those already accessed for the link to arrive at a total number of channels.

The FIRST-WORK-FILE is then opened and read. If the Input Event file link number is equal to the TRANSLATE-TABLES file link number, an additional test is performed to determine if the channel number of the event has exceeded the number of channels indicated by the TRANSLATE-TABLES file. If so, the TRANSLATE-TABLES file is re-read and the matching channel speed group made available. If the channel speed group has not changed, input data on the record is moved to an output area preparatory for release to the SORT. If this record (the initial event record) is a start of message (SOM), the record time is established as the SOM-TIME, overriding the initialized value set at START-TIME. If the record is a restoral (ORAC=9), an assumption is made that this link/channel has been out since START-TIME and the outage time is determined by subtracting the record time from START-TIME. If the record is an end of message (EOM), an addition to the link/channel active time is made in the amount of the difference between the record time and START-TIME. If the record is a message interrupt (ORAC=7), it is bypassed since the active time is based on the initialized START-TIME. If the record is an outage (ORAC=8), the record time is set as the TIME-OF-OUTAGE and a scanning loop is entered to determine the restoral time. When this occurs, the link outage time is augmented by the difference between the TIME-OF-OUTAGE and the restoral record time.

After any of these activities, control is passed to a Main-Loop for processing subsequent events. Whenever a change in the message number occurs, the assumption is a new SOM, and the SOM-TIME is set accordingly. EOM events cause augmentation of the line-block and completed message counters for each level of precedence in addition to the active time counter.

Message interrupts (ORAC=7) also augment line-block and active time counters. Outages transfer control to the restoral scanning loop.

Changes in the channel speed group cause computation of the capacity and utilization factor for the channel speed group, movement of the necessary TRANSLATE-TABLES file data to the output area, and release of data to the SORT. The utilization is computed by the following formula:

$$\text{LINK/CHANNEL UTILIZATION} = \frac{\text{TOTAL LINE BLOCKS 100}}{\sum \text{CHANNEL SPEED GROUP CAPACITY}}$$

The record is then released to the final SORT, work areas are reinitialized, and the entire process is repeated until the last input record has been processed.

The final SORT returns the file to the output procedure print routines (LINK-UTL-PRINT Section) in translated line, source switch, and destination switch name sequence. As these records are returned, comparison of the current record's link name is made to a store field containing the link name of the previous record. If the names match, the input (SORT-3-RECORD) record is placed in a work area (LINK-PERIOD), the work area is moved to a print area (LINK-LINE-1, LINK-LINE-2), printed, and another record is read. If the names do not match, an additional test is made to determine if both directions of the previous link were displayed. If only the first direction has printed, a message is generated stating that traffic for the second direction was not presented to the Reports program. After that message, or if both directions did print, page break headers are printed, the previous name store field is reset, and printing is performed. This process is repeated until all records have been processed and control is then returned to the Root Segment via the base routines.

7.4.2.2 Group 2 Reports

This group is comprised of the Link, Switch, and Tributary Queue Reports. As with the Group 1 reports, these reports are generated by a SORT statement with a common input procedure to prepare the data and unique output procedures to print the prepared data. In addition, the Translation Tables generated by the Status Generator are used to translate link, switch, and tributary numbers to their model names.

7.4.2.2.1 Input Procedure

Prior to issuing the SORT statement, the Translation Tables are loaded. Depending on which report is requested, data is read into either the XLATE-SWITCH-TABLE, XLATE-TRIB-TABLE, or XLATE-LINK-TABLE areas of Working Storage. If the Link or Switch card options were present, only those Translation records with names that match the desired links or switches are allowed to load. When table loading is complete, control passes to the respective SORT statement for each report.

The common input procedure (BUILD-QUEUE-FILE) performs time scanning of the STATUS-FILE to select only those records within the time frame established by the Start and Stop card options. If neither of these options is specified, the entire Status file is processed with the first and last clock records used as the start and stop times.

For Link and Switch Queue requests, screening is performed to pass only those records matching the link or switch numbers in the previously loaded Translation Tables. Tributary Queue requests are not screened in this manner, but are passed directly to the Sort Release routine.

The Sort Release routine includes the provision to group up to seven Link Queue records or up to three Switch Queue records in each sort record. The size of these groups is predicated on the number of individual link or switch columns that may be displayed within the 132 print positions of the output reports. Tributary Queue records are not grouped, but released individually to the SORT.

Upon completion of these processes, control passes to the SORT statement. The files for the Link and Switch Queue reports are sorted in time and FIRST-ELEMENT-ADDRESS sequence and the Tributary file by address and time.

7.4.2.2.2 Print Link Queue Report Output Procedure

To this point the Link Queue Report file has been screened to eliminate unwanted data and sorted into the proper report sequence. The remaining steps performed by this segment are to translate the link address and its terminating switch addresses to their user designations, select those records specified by the Queue Sample Index, format the data, and print the report.

Each record read by the Print segment is tested for the number of individual link records grouped in the SORT record. If there are less than seven, a COLUMN-COUNT is set to eliminate printing of all but that number of columns. The high and low precedence queue values are then summed and compared to a MAX-Q-SIZE-TABLE to determine the point at which the link's queue was greatest. Any queue size exceeding the previously stored value is moved to the MAX-Q-SIZE-TABLE and the time of the maximum moved to a MAX-Q-TIME-TABLE, both under control of a subscript corresponding to the position of the link record within the sort record grouping. Control then passes to the LINK-QUEUE-SAMPLE-INDEX-TEST where a check is made to allow further processing of only those records corresponding to the Sample Index Value, if it is present.

Each record passing the LINK-QUEUE-SAMPLE-INDEX-TEST is then moved to a format area and printed. Control then returns to the input routine to obtain another sorted record.

As each successive record is returned from the SORT, comparison of the first link's address in the group is made to a store area (THE-LAST-LINK). When a change in this address is detected, a routine is performed to print the maximum queue size and its time of occurrence and establish report headings for the next group of links.

The initial record of each new group is used to obtain the translated link and switch designators from the Translation Tables. After the report headers are printed, the initial record and subsequent records are processed. After all records have printed, the Print segment is exited and control returned to the Root Segment via the base routine.

7.4.2.2.3 Print Switch Queue Report Output Procedure

As with the Link Queue Report segment, the Switch Queue Report segment translates the switch address, selects only those records specified by the Queue Sample Index, formats the data, and prints the Switch Queue Report. Each sort record read is examined for the number of switch records in the group. If less than three, the COLUMN-COUNT is set accordingly and excess switch columns are eliminated.

Each of the three queues for each switch (TRIB-IN, TRIB-OUT, and LINK) is compared to a queue-size table for determination of the size and time when each is greatest. A queue sample index test is made to pass only those records corresponding to the index value which are then formatted and printed. Initial switch records in each sort record group are used to translate switch addresses, cause printing of maximum queue lines when each group has ended, and print headers for the next group. Upon completion of the report, control is returned to the Root Segment via the base routine.

7.4.2.2.4 Tributary Queue Report Output Procedure

The remaining steps necessary to produce the Tributary Queue Report are to examine the queue sizes for each tributary, select the largest queue size and the time that it occurred, and print this data.

Since the SORT has placed the incoming data in sequence by tributary address, each record's input and output queue sizes are compared against a table of queue sizes initialized at zero. Whenever a queue size exceeds the table value, the records queue size and time are placed in the table and another record read. This process continues until a change in the tributary address occurs. Each change of address causes augmentation of a subscript used in conjunction with the queue size and time tables. Maximum values for each tributary queue are held in these tables until the subscript reaches a value determined by the number of tributary records that can be formatted within the 132 print positions. When the subscript reaches this value, translation tables are searched to convert the tributary address to their model designations. These designations are formatted with the associated maximum queue sizes and times, the tables are cleared, and processing begins for another line. After all records have been processed, control is returned to the Root Segment via the base routine.

Program logic flow charts are included at the end of this section.

7.5 MACHINE DEPENDENCE

Two elements of the Reports program are not compatible with the spectrum of manufacturers subscribing to ANSI COBOL. The first element of dependency in the IBM 360/OS consists of the Job Control Language (JCL) statements related to data management and job execution. If Reports program is implemented in an environment other than the IBM 360/OS, the JCL statements must be adapted to their respective environment. The second element of dependency is that in several sections the TRANSFORM verb (an IBM extension of ANSI COBOL) was used to reduce run time. These departures have been identified in the source language code edit by an IBM EXT entry in Columns 73-80. COBOL environments not supporting the TRANSFORM verb will require program modification and re-compilation. The programmer merely need replace the TRANSFORM statements with EXAMINE/REPLACING statements for identical operation.

No discernible hardware dependencies apply to the Reports program.

Table 7-1. Option Run Card - Type Report Card

<table>
<tr><th>CARD COLUMN</th><th>DESCRIPTION</th><th>EXPLANATION</th></tr>
<tr><td>1-12</td><td>Report Name</td><td>The report to be produced is controlled by this entry. Coding is as follows:
<table>
<tr><th>Report Produced</th><th>Must Begin in CC-1</th></tr>
<tr><td>*Speed of Service</td><td>SOS REPORT</td></tr>
<tr><td>Link Utilization</td><td>LINK UTILITY</td></tr>
<tr><td>In-Station Handling</td><td>IN STATION</td></tr>
<tr><td>Link Queue</td><td>LINK QUEUE</td></tr>
<tr><td>Switch Queue</td><td>SWITCH QUEUE</td></tr>
<tr><td>Trib Queue</td><td>TRIB QUEUE</td></tr>
<tr><td>Network Delay</td><td>NET DELAY</td></tr>
</table></td></tr>
<tr><td>13-80</td><td>Not used</td><td>Comments may be entered in this space.
*Will also produce Average SOS Report when either the All Switches or Switch Pairs options are used.</td></tr>
</table>

Table 7-2. Option Run Card - Time Cards

CARD COLUMN	DESCRIPTION	EXPLANATION
1-6	Start Time Identifier	Identifies this card as containing the beginning time of period to be examined for any report. Any event or status record occurring prior to time indicated will not be processed by Reports program. Coding must begin in cc-1 and continue as follows. START,
7	Blank	
8-11	Start Time	Starting time in military time format, i.e., HHMM. Leading zeros are required. Range of values is 0000 - 3600. Value must be less than value presented by Stop card, if used.
12-80	Not used	Comments may be entered in this space.
1-5	Stop Time Identifier	Identifies this card as containing the end time of period to be examined for any report. Any event or status record occurring after time indicated will not be processed by Reports program. Coding must begin in cc-1 and continue as follows: STOP,
6	Blank	
7-10	Stop Time	Ending time in military time format, i.e., HHMM. Leading zeros are required. Range of values is 0000 - 3600. Value must be greater than value presented by Start card, if used.
11-80	Not used	Comments may be entered in this space.

Table 7-3. Option Run Card - LMF Card

CARD COLUMN	DESCRIPTION	EXPLANATION
1-4	LMF Indicator	Identifies this card as containing Language Media Format to be selected for processing. If no LMF card is present, all media formats will process. Coding must begin in cc-1 and continue as follows: LMF =
5	LMF Code	Contains one of the following values: LMF Code Interpolation 1 TTY 2 Punched Card 3 Magnetic Tape 4 Miscellaneous
6-80	Not used	Comments may be entered in this space.

Table 7-4. Option Run Card - Queue Sample Index Card

CARD COLUMN	DESCRIPTION	EXPLANATION
1-5	Sample Index Indicator	Identifies this card as containing the Link or Switch Queue Sampling Index, i.e., only each Nth record will print at report time. Coding must begin in cc-1 and continue as follows: TIME =
6-9	Sample Index	Range of values presented may be between 1 and 9999, right justified. Leading zeroes not required.
10-80	Not used	Comments may be entered in this space.

Table 7-5. Option Run Card - Interval Card

CARD COLUMN	DESCRIPTION	EXPLANATION
1-10	Interval Card Identifier	First of two possible cards containing the upper-class limits used in presenting SOS, Network Delay, or In-Station-Handling times. Coding must begin in cc-1 and continue as follows: INTERVALS,
11	Blank	
12-80	New Intervals	New upper-class limit values. Each new value must be separated by a comma and the last value must be immediately followed by ($). Maximum value usable is 99999. Coding is free form in that leading zeros and space separators between values are not required Number of values presented may vary from 1 to 16. Non-numeric entries other than comma, $, or blank will cause unpredictable results.
1-80	(Card 2)	If continuation to a second card is required, the last value on Card 1 may overlap on to Card 2. Coding on Card 2 follows the same conventions as cc 12-80 on Card 1. Note: Only two INTERVAL cards may be submitted in the same request, must be in the indicated sequence, and must be contiguous.

Table 7-6. Option Run Card - All Switches Card

CARD COLUMN	DESCRIPTION	EXPLANATION
1-12	All Switches Indicator	Establishes necessary controls to determine the SOS between switches. Coding must begin in cc-1 and continue as follows: ALL SWITCHES
13-80	Not used	Comments may be entered in this space.

Table 7-7. Option Run Card - Switch Pair Card

CARD COLUMN	DESCRIPTION	EXPLANATION
1-8	Switch Pair Indicator	Contains a source and destination switch combination. Traffic between the two switches, in the indicated direction, will be selected for SOS processing. No other data will be processed. A Switch Pair card is required for each combination to be processed. Coding must begin in cc-1 and continue as follows: SWITCH (
9-14	Source Switch	Contains the source switch designation, left justified.
15	Source Switch Delimitor	Should contain the right parenthesis character).
16-23	Blank	
24	Destination Switch Delimitor	Should contain the left parenthesis character (.
25-30	Destination Switch	Contains the destination switch designation, left justified.
31	Destination Switch Delimitor	Should contain the right parenthesis character).
32-80	Not used	Comments may be entered in this space.

Table 7-8. Option Run Card - All Tribs Card

CARD COLUMN	DESCRIPTION	EXPLANATION
1-9	All Tribs Indicator	Establishes necessary controls to determine SOS between tributaries. Coding must begin in cc-1 and continue as follows: ALL TRIBS
10-12	Blank	
13-80	Not used	Comments may be entered in this space.

Table 7-9. Option Run Card - Trib Pair Card

CARD COLUMN	DESCRIPTION	EXPLANATION
1-6	Trib Pair Indicator	Contains a source and destination tributary combination. Traffic between them, in the indicated direction, will be selected for SOS processing. No other data will be processed. A Trib Pair card is required for each combination to be processed. Coding must begin in cc-1 and continue as follows: TRIB (
7-12	Source Tributary	Contains source tributary designator, left justified.
13	Source Tributary Delimitor	Should contain right parenthesis character).
14-20	Blank	
21	Destination Tributary Delimitor	Should contain left parenthesis character (.
22-27	Destination Tributary	Contains destination tributary designator, left justified.
28	Destination Tributary Delimitor	Should contain right parenthesis character).
29-80	Not used	Comments may be entered in this space.

Table 7-10. Option Run Card - Link Card/Switch Card

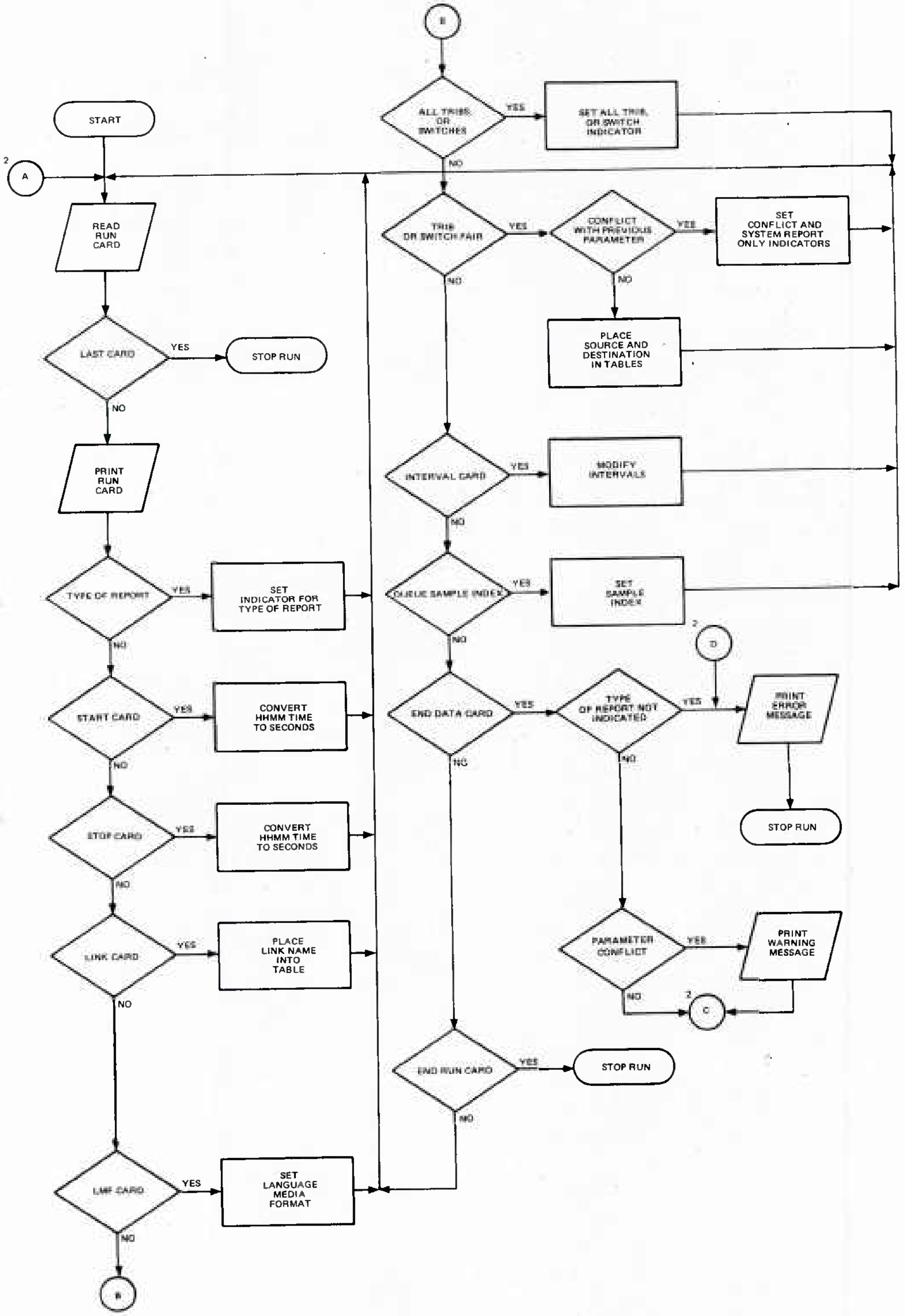
CARD COLUMN	DESCRIPTION	EXPLANATION
1-4	Link Indicator	Contains an individual link designator that will be selected for processing by Link Queue Report processor. Coding must begin in cc-1 and continue as follows: LINK
5	Blank	
6-11	Link Designator	Contains link designator, left justified.
12-80	Not used	Comments may be entered in this space.
1-6	Switch Indicator	Contains an individual designator that will be selected for processing by the Switch Queue Report routine. Coding must begin in cc-1 and continue as follows: SWITCH
7	Blank	
8-13	Switch Designator	Contains Switch designator, left justified.
14-80	Not used	Comments may be entered in this space.

Table 7-11. Option Run Card - End of Request

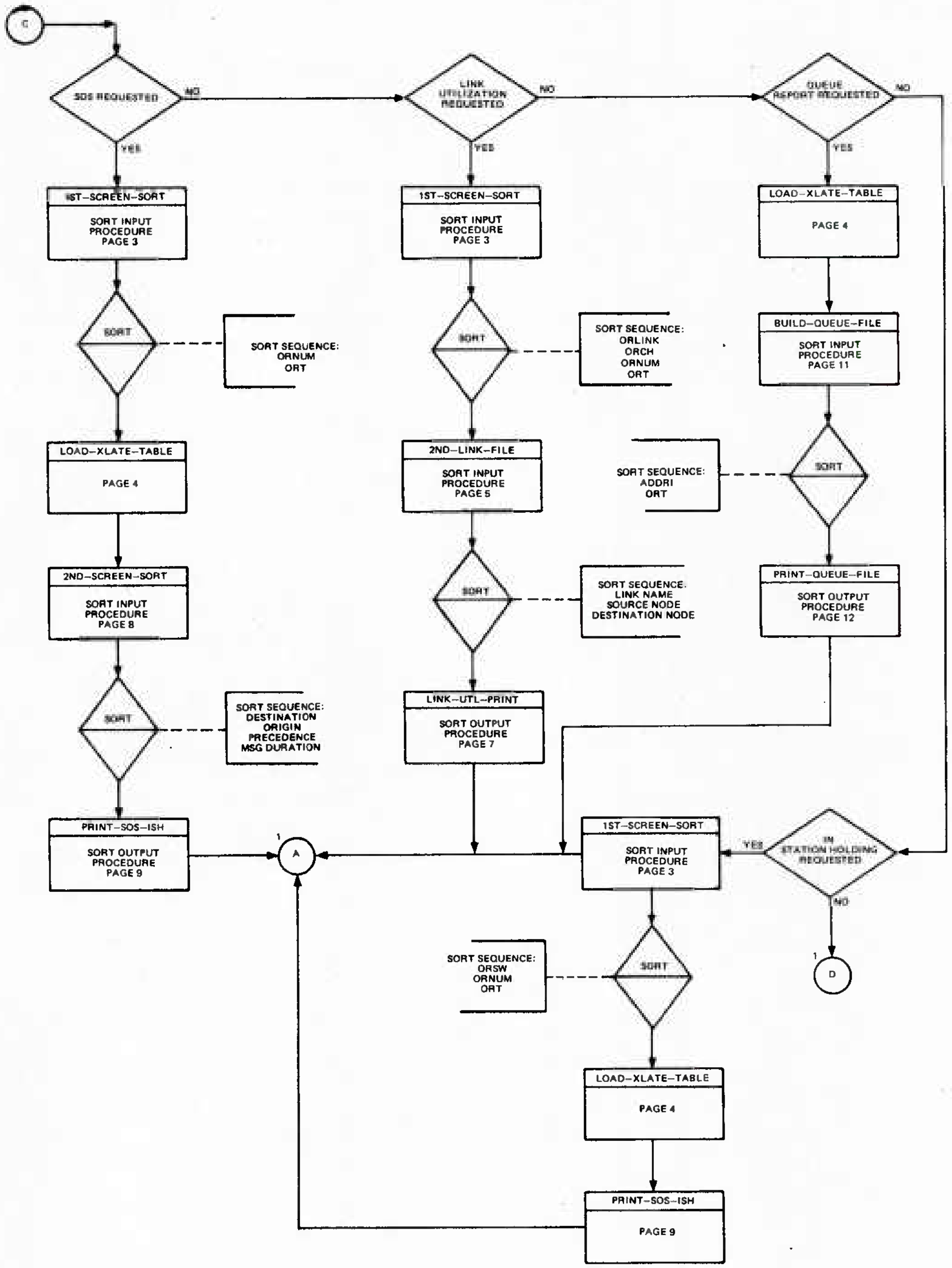
CARD COLUMN	DESCRIPTION	EXPLANATION
1-8	End of Individual Report Request	Marks end of a set of Run cards for a particular report request. Coding must begin in cc-1 and continue as follows: END DATA
9-12	Blank	
13-80	Not used	Comments may be entered in this space.

Table 7-12. Option Run Card Summary

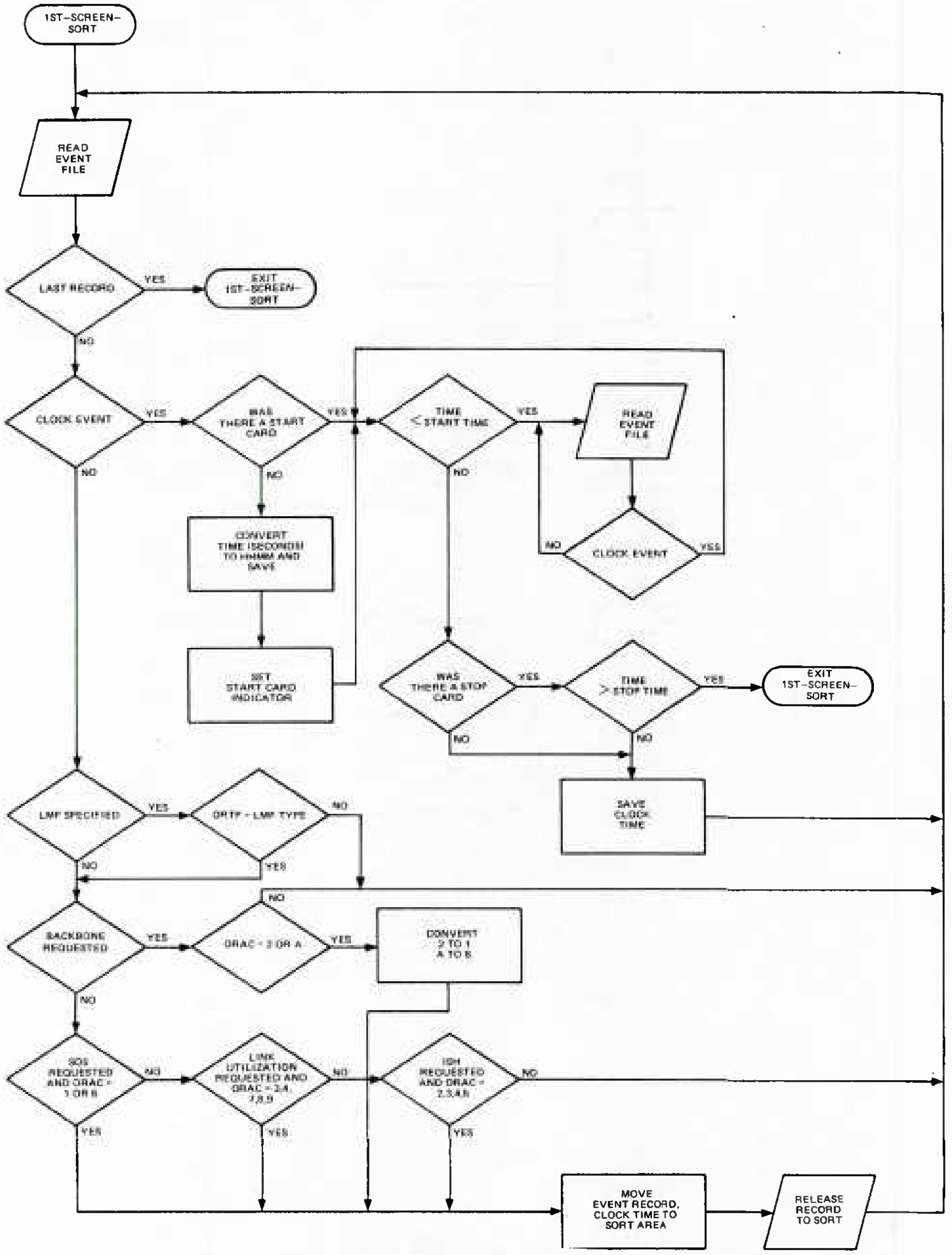
Report Option Card	Network Delay and Speed of Service	In Station Holding Time	Link Utilization	Link Queue	Switch Queue	Trib Queue
Type of Report	Required	Required	Required	Required	Required	Required
Start Time	Optional	Optional	Optional	Optional	Optional	Optional
Stop Time	Optional	Optional	Optional	Optional	Optional	Optional
LMF	Optional	Optional	Optional	N/A	N/A	N/A
All Switches	Optional	N/A	N/A	N/A	N/A	N/A
All Tribs	Optional	N/A	N/A	N/A	N/A	N/A
Switch Pairs	Optional	N/A	N/A	N/A	N/A	N/A
Trib Pairs	Optional	N/A	N/A	N/A	N/A	N/A
Link	N/A	N/A	N/A	Optional	N/A	N/A
Switch	N/A	N/A	N/A	N/A	Optional	N/A
Interval	Optional	Optional	N/A	N/A	N/A	N/A
Sample Index	N/A	N/A	N/A	Optional	Optional	N/A
End Data	Required	Required	Required	Required	Required	Required



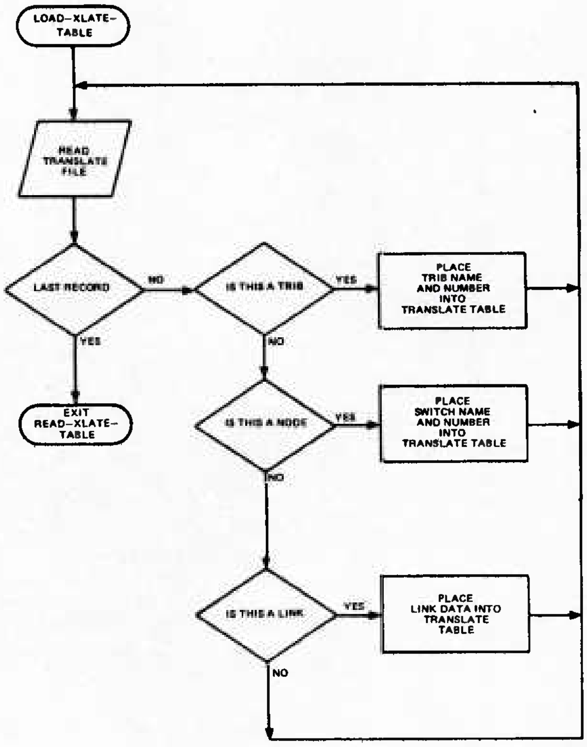
Reports Program (Sheet 1 of 12)


C
SOS REQUESTED
NO
YES
1ST–SCREEN–SORT
SORT INPUT PROCEDURE PAGE 3
SORT
SORT SEQUENCE: ORNUM ORT
LOAD–XLATE–TABLE
PAGE 4
2ND–SCREEN–SORT
SORT INPUT PROCEDURE PAGE 8
SORT
SORT SEQUENCE: DESTINATION ORIGIN PRECEDENCE MSG DURATION
PRINT–SOS–ISH
SORT OUTPUT PROCEDURE PAGE 9
1
A
LINK UTILIZATION REQUESTED
NO
YES
1ST–SCREEN–SORT
SORT INPUT PROCEDURE PAGE 3
SORT
SORT SEQUENCE: ORLINK ORCH ORNUM ORT
2ND–LINK–FILE
SORT INPUT PROCEDURE PAGE 5
SORT
SORT SEQUENCE: LINK NAME SOURCE NODE DESTINATION NODE
LINK–UTL–PRINT
SORT OUTPUT PROCEDURE PAGE 7
QUEUE REPORT REQUESTED
NO
YES
LOAD–XLATE–TABLE
PAGE 4
BUILD–QUEUE–FILE
SORT INPUT PROCEDURE PAGE 11
SORT
SORT SEQUENCE: ADDRI ORT
PRINT–QUEUE–FILE
SORT OUTPUT PROCEDURE PAGE 12
IN STATION HOLDING REQUESTED
YES
NO
1
D
1ST–SCREEN–SORT
SORT INPUT PROCEDURE PAGE 3
SORT
SORT SEQUENCE: ORSW ORNUM ORT
LOAD–XLATE–TABLE
PAGE 4
PRINT–SOS–ISH
PAGE 9

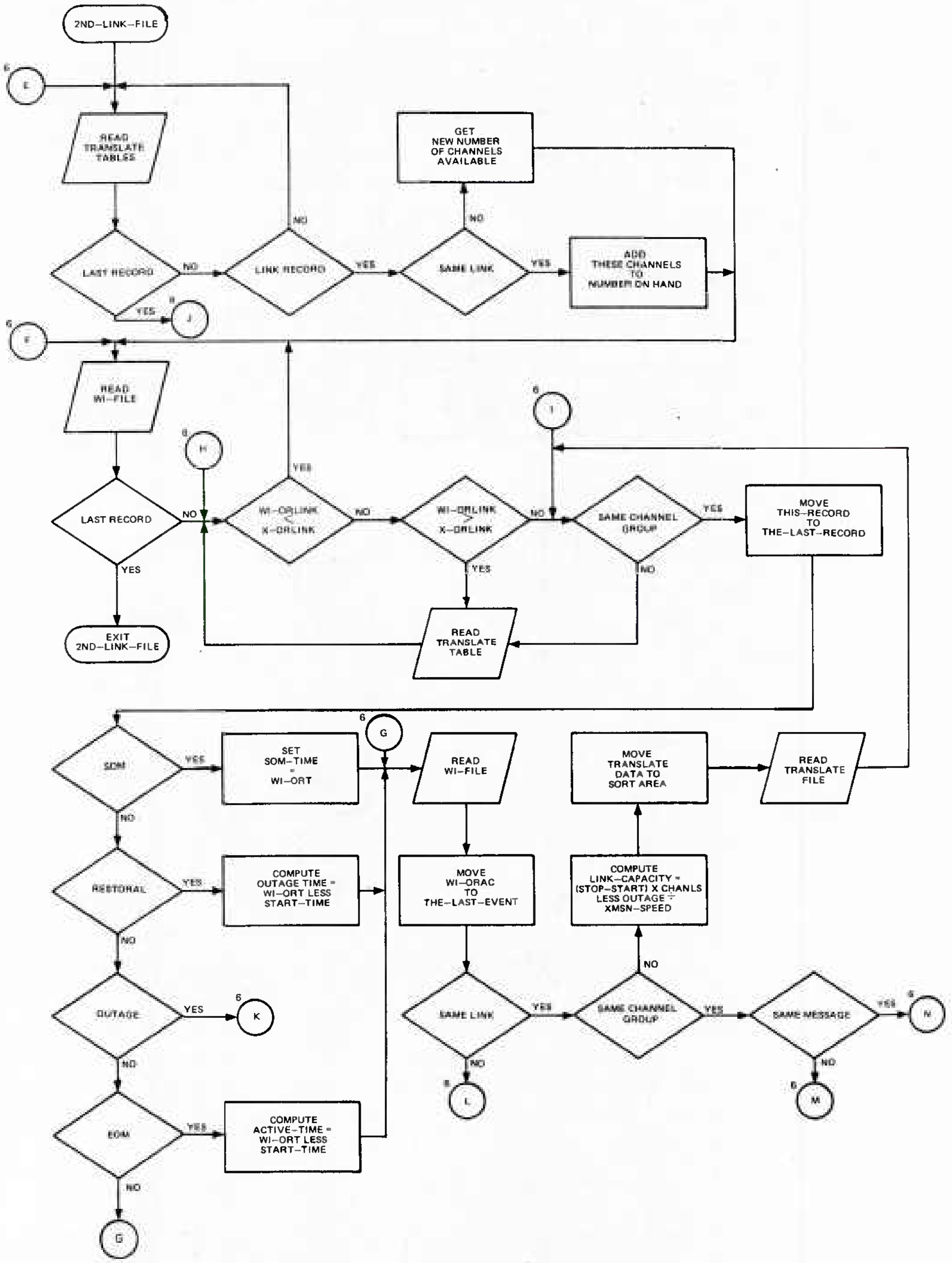
Reports Program (Sheet 2 of 12)



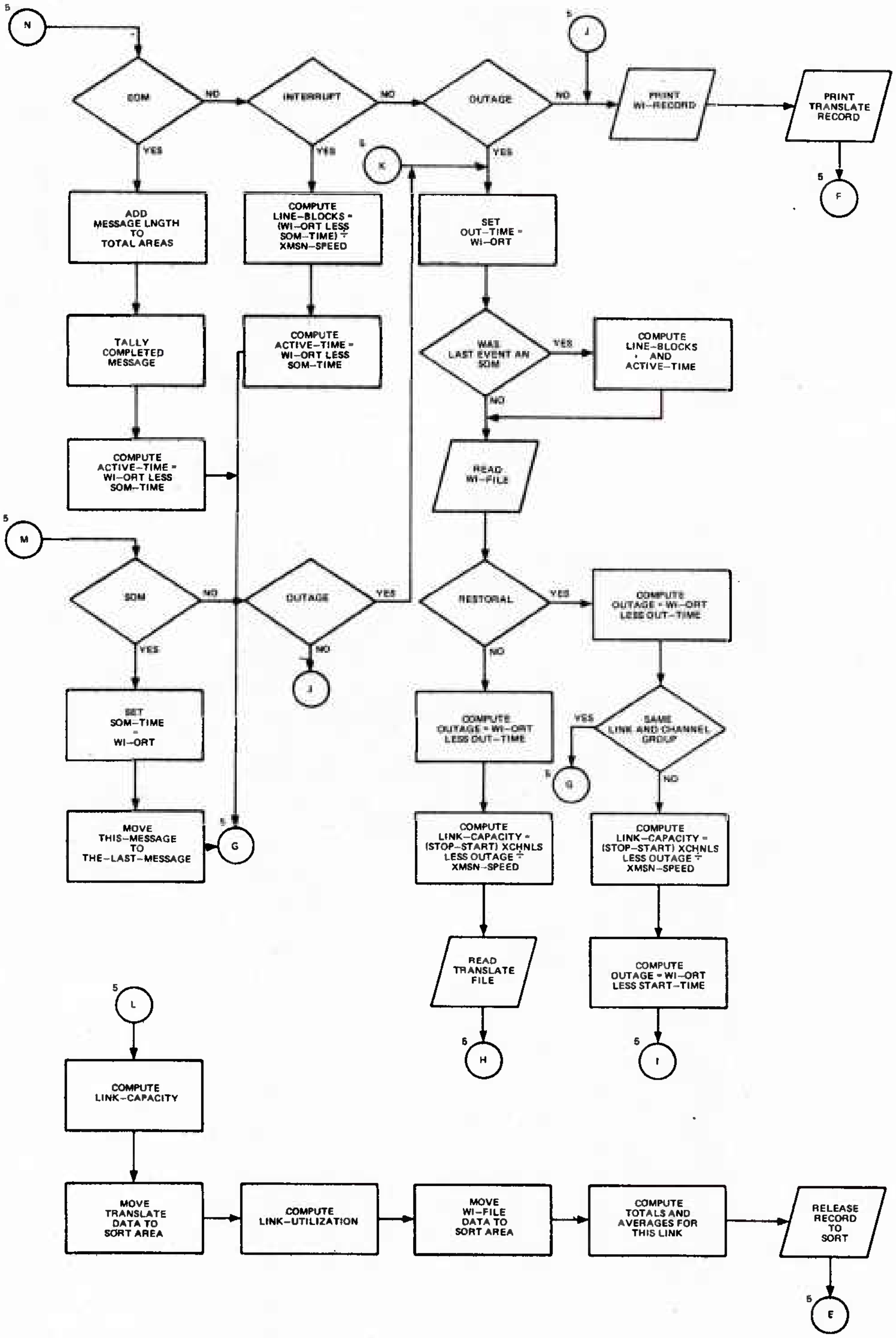
Reports Program (Sheet 3 of 12)


LOAD–XLATE–TABLE
READ TRANSLATE FILE
LAST RECORD
NO
YES
EXIT READ–XLATE–TABLE
IS THIS A TRIB
YES
PLACE TRIB NAME AND NUMBER INTO TRANSLATE TABLE
NO
IS THIS A NODE
YES
PLACE SWITCH NAME AND NUMBER INTO TRANSLATE TABLE
NO
IS THIS A LINK
YES
PLACE LINK DATA INTO TRANSLATE TABLE
NO

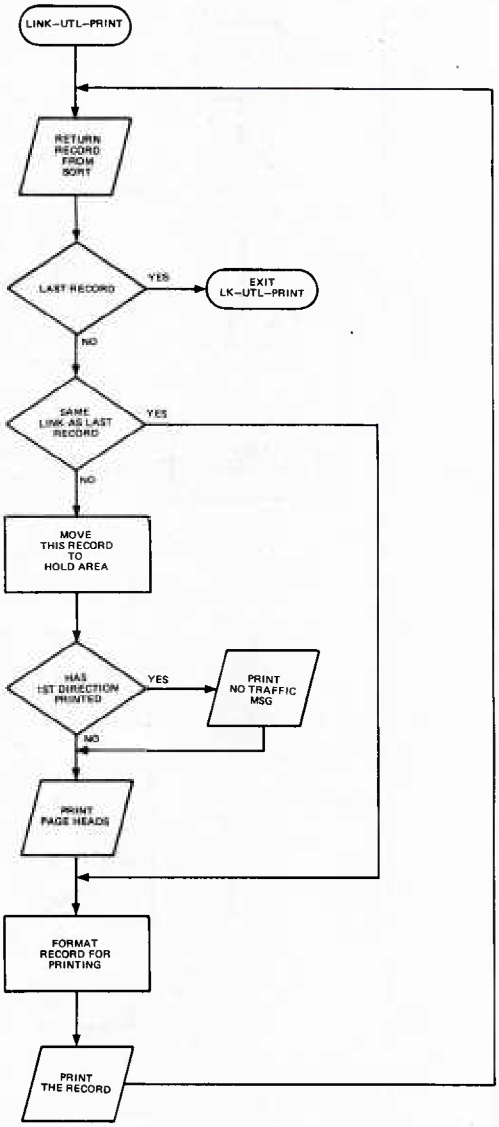
Reports Program (Sheet 4 of 12)



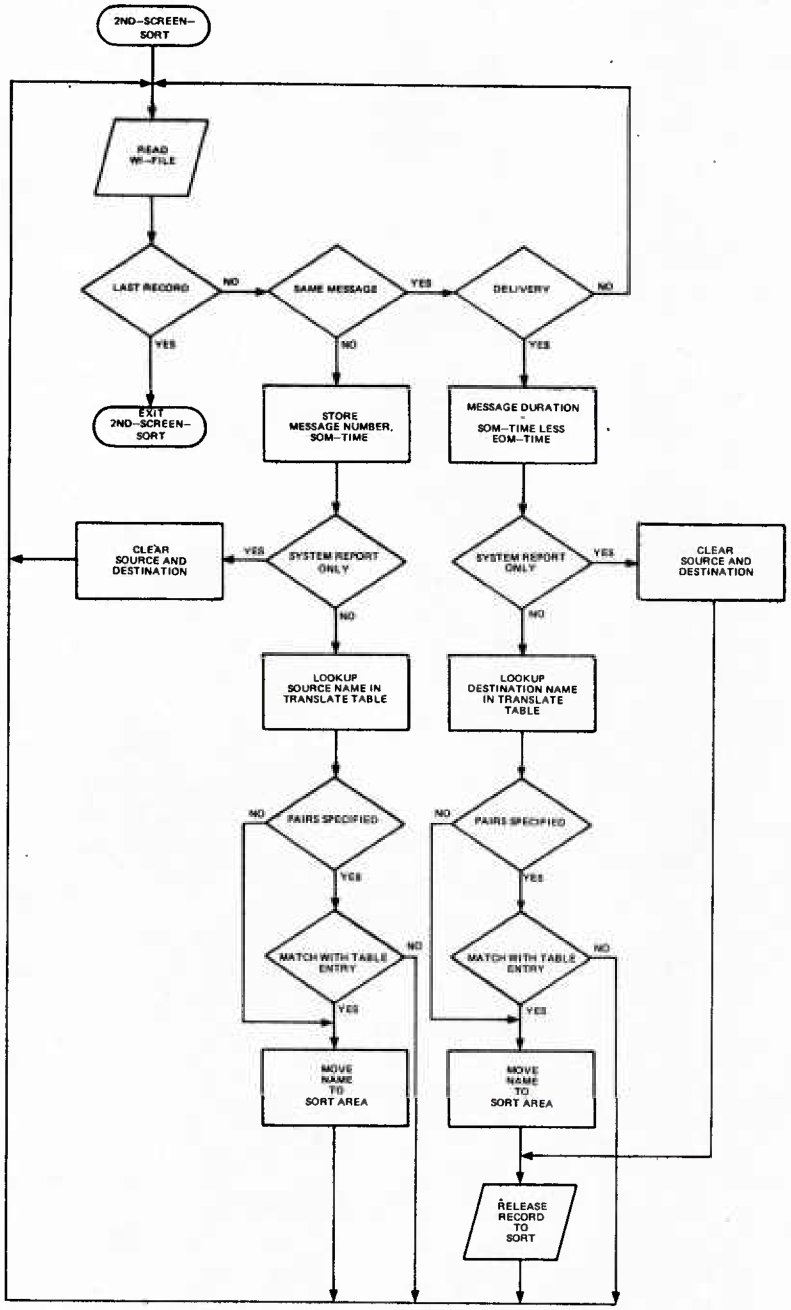
Reports Program (Sheet 5 of 12)


5
N
SOM
NO
YES
ADD MESSAGE LNGTH TO TOTAL AREAS
TALLY COMPLETED MESSAGE
COMPUTE ACTIVE-TIME = WI-ORT LESS SOM-TIME
INTERRUPT
NO
YES
COMPUTE LINE-BLOCKS = (WI-ORT LESS SOM-TIME) ÷ XMSN-SPEED
COMPUTE ACTIVE-TIME = WI-ORT LESS SOM-TIME
5
K
OUTAGE
NO
YES
SET OUT-TIME = WI-ORT
WAS LAST EVENT AN SOM
YES
NO
COMPUTE LINE-BLOCKS , AND ACTIVE-TIME
READ WI-FILE
5
J
PRINT WI-RECORD
PRINT TRANSLATE RECORD
5
F
5
M
SOM
NO
YES
SET SOM-TIME = WI-ORT
MOVE THIS-MESSAGE TO THE-LAST-MESSAGE
5
G
OUTAGE
YES
NO
J
RESTORAL
YES
NO
COMPUTE OUTAGE = WI-ORT LESS OUT-TIME
COMPUTE OUTAGE = WI-ORT LESS OUT-TIME
SAME LINK AND CHANNEL GROUP
YES
NO
5
G
COMPUTE LINK-CAPACITY = (STOP-START) XCHNLS LESS OUTAGE ÷ XMSN-SPEED
COMPUTE LINK-CAPACITY = (STOP-START) XCHNLS LESS OUTAGE ÷ XMSN-SPEED
READ TRANSLATE FILE
5
H
COMPUTE OUTAGE = WI-ORT LESS START-TIME
5
I
5
L
COMPUTE LINK-CAPACITY
MOVE TRANSLATE DATA TO SORT AREA
COMPUTE LINK-UTILIZATION
MOVE WI-FILE DATA TO SORT AREA
COMPUTE TOTALS AND AVERAGES FOR THIS LINK
RELEASE RECORD TO SORT
5
E

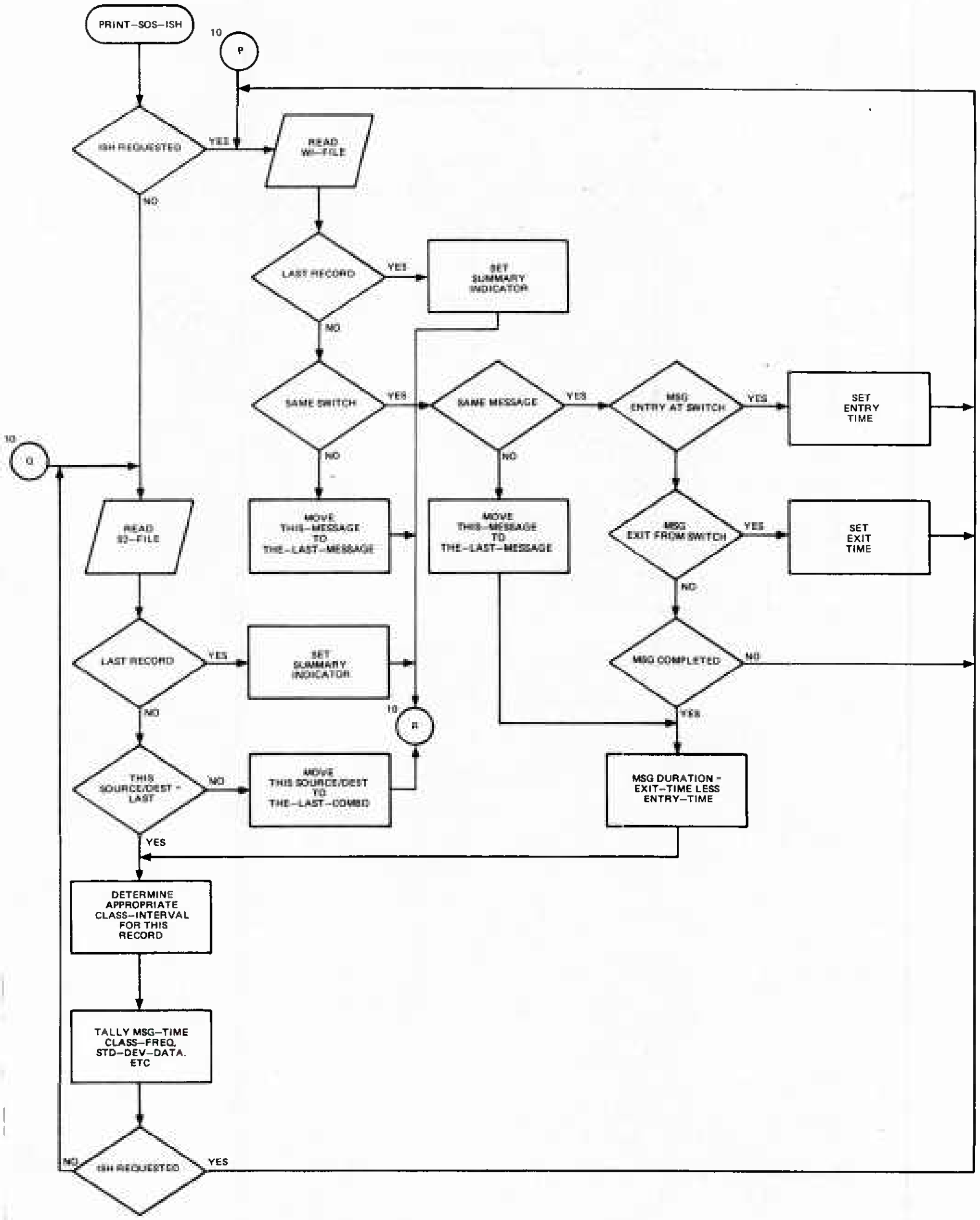
Reports Program (Sheet 6 of 12)



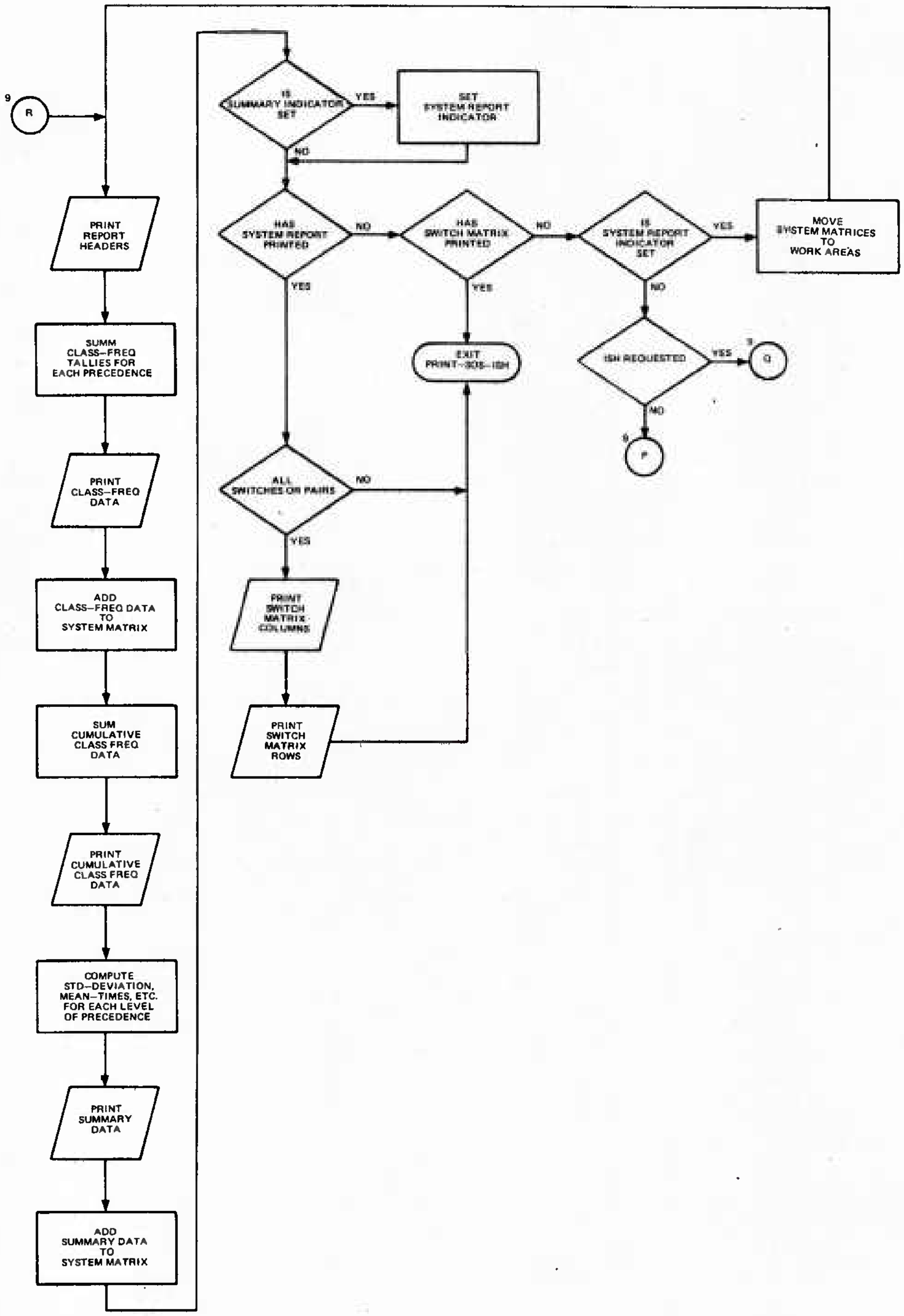
Reports Program (Sheet 7 of 12)



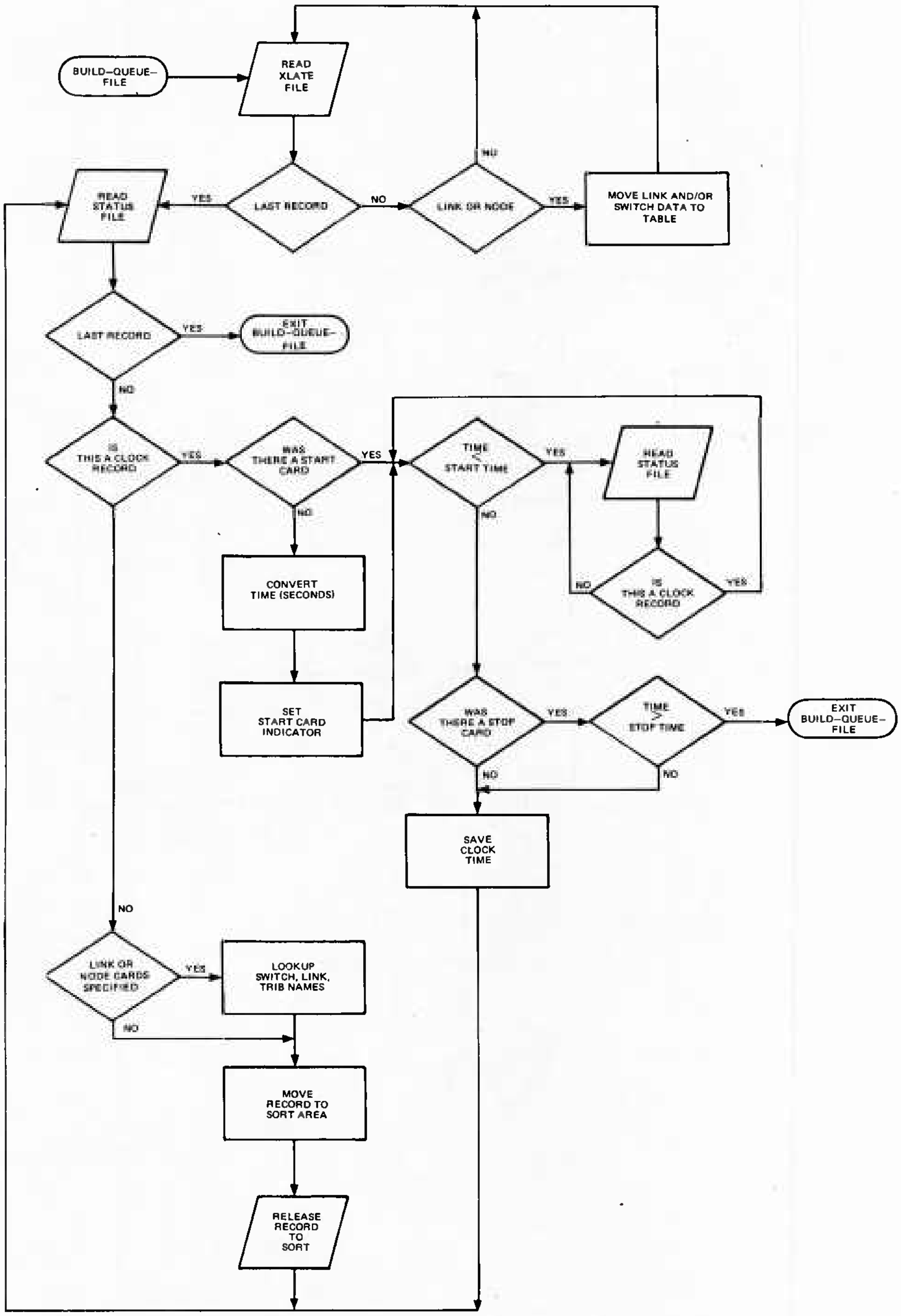
Reports Program (Sheet 8 of 12)



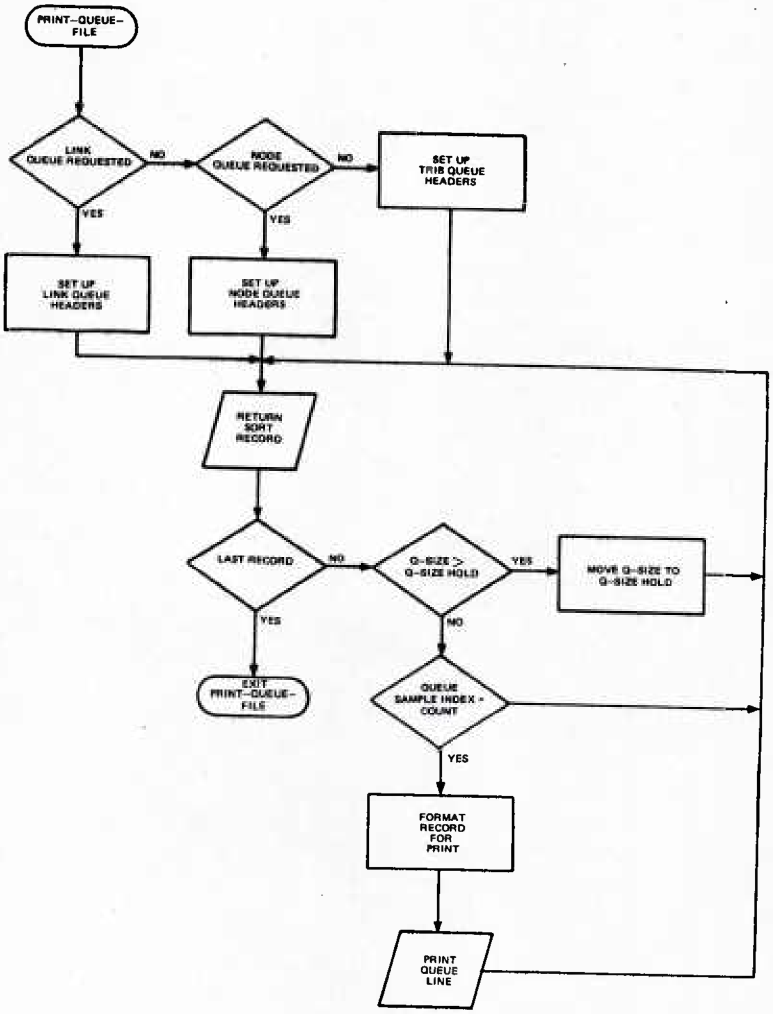
Reports Program (Sheet 9 of 12)


9
R
PRINT REPORT HEADERS
SUMM CLASS–FREQ TALLIES FOR EACH PRECEDENCE
PRINT CLASS–FREQ DATA
ADD CLASS–FREQ DATA TO SYSTEM MATRIX
SUM CUMULATIVE CLASS FREQ DATA
PRINT CUMULATIVE CLASS FREQ DATA
COMPUTE STD–DEVIATION, MEAN–TIMES, ETC. FOR EACH LEVEL OF PRECEDENCE
PRINT SUMMARY DATA
ADD SUMMARY DATA TO SYSTEM MATRIX
IS SUMMARY INDICATOR SET
YES
SET SYSTEM REPORT INDICATOR
NO
HAS SYSTEM REPORT PRINTED
NO
HAS SWITCH MATRIX PRINTED
NO
IS SYSTEM REPORT INDICATOR SET
YES
MOVE SYSTEM MATRICES TO WORK AREAS
YES
YES
EXIT PRINT–SOS–ISH
NO
ISH REQUESTED
YES
9
Q
NO
9
P
ALL SWITCHES OR PAIRS
NO
YES
PRINT SWITCH MATRIX COLUMNS
PRINT SWITCH MATRIX ROWS

Reports Program (Sheet 10 of 12)



Reports Program (Sheet 11 of 12)



Reports Program (Sheet 12 of 12)

SECTION 8 - LIST EVENTS PROGRAM

8.1 PROGRAM REQUIREMENT DESCRIPTION

8.1.1 Program Name: List Events

8.1.2 Program Identification: DNLSTEV1

8.1.3 Purpose

Each major event which occurs within the Basic Simulation program is recorded by writing an event descriptor record on one of the output files. Similarly, the status of the network being simulated is reflected by writing status descriptor records on an output file at specified time intervals. The List Events program provides a detailed listing of the event and status records produced by the Basic Simulation program. Input parameter cards are used to filter the records to reduce the listing to a given area of interest.

8.1.4 Requirements

The program is compiled in FORTRAN IV and requires approximately 110K bytes of core when executed on an IBM 360/65.

8.2 INPUT SPECIFICATIONS

The program inputs are four data files generated by previously executed programs and a control deck specifying the desired options.

8.2.1 Event Record Inputs

The events which have taken place in the Basic Simulation program are written on the Speed of Service (SOS) Event and Link Event files. Generation of these files is described in Paragraph 6.3.

8.2.2 Status Record Inputs

The status of the network being simulated is noted by writing Switch, Link, and Tributary Status Records at specified intervals. Generation of the Status Record file is described in Paragraph 6.3.

8.2.3 Translation File

The Translation file is a required input to print the names of network elements. The generation of this file is described in Paragraph 2.3.

8.2.4 Control Parameters

Thirteen different types of parameter cards are used to define areas of interest. Table 8-1 gives the format of these parameter cards.

8.3 OUTPUT SPECIFICATIONS

The program output is a listing of selected records (as determined by parameter cards). The listing may have up to four parts: Switch Status, Link Status, Tributary Status, and Event Records.

8.4 PROGRAM LOGIC

8.4.1 Input Parameter Logic

The input parameter define the types of records to be printed. For the status records, the print record options are:

1. All status records
2. All switch records
3. All link records
4. All tributary records
5. Switch records for specified switches
6. Link records for specified links
7. Tributary records for specified tributaries.

The status records may be printed when the queues described in the records equal or exceed a given value. For the Event records the print record options are:

1. All event records
2. All switch records
3. All link records

4. All tributary records
5. Switch records for specified switches
6. Link records for specified links
7. Tributary records for specified tributaries
8. All records containing specified message numbers
9. All records of specified action codes.

In addition, in all parameter cards, except those for status queues, a start and stop time is given, establishing a time frame for the selection.

When each parameter card is read into the program, a flag indicating the type of selection is set and, where appropriate, the time frame entries are stored. Where switch, link and tributary names are used, the address identification is retrieved from the Translation Tables and stored into an appropriate array, with specified message numbers and action codes are also stored. When status queues are specified, the desired value is stored.

8.4.2 Program Execution

The program first reads the desired elements from the Tranlation file into switch, link, and tributary arrays. The parameter cards are read, setting flags, filing arrays, etcetera. Status flags are checked to seek if status listings are desired. If any flags are set, the file is read. As each record is read in, the type of record (switch, link, or tributary) is determined and the program branches to the appropriate section where it determines if that record meets the selection criteria. If not, the next record is checked. If the record is selected, and is a switch record, it is printed. Due to a difference in format, link and tributary records are stored on temporary files and printed when the end of the Status file is reached.

Since there are two overlaping event files, each in chronological sequence, a read-merge type operation is performed selecting the next record in chronological order. This record is checked to see if it meets the selection criteria and, if it does, it is printed.

Program logic flow charts are included at the end of this section.

8.4.3 Memory Layout

The program does not use subroutines and requires a single block of memory for executable instructions and storage arrays.

8.5 MACHINE DEPENDENCE

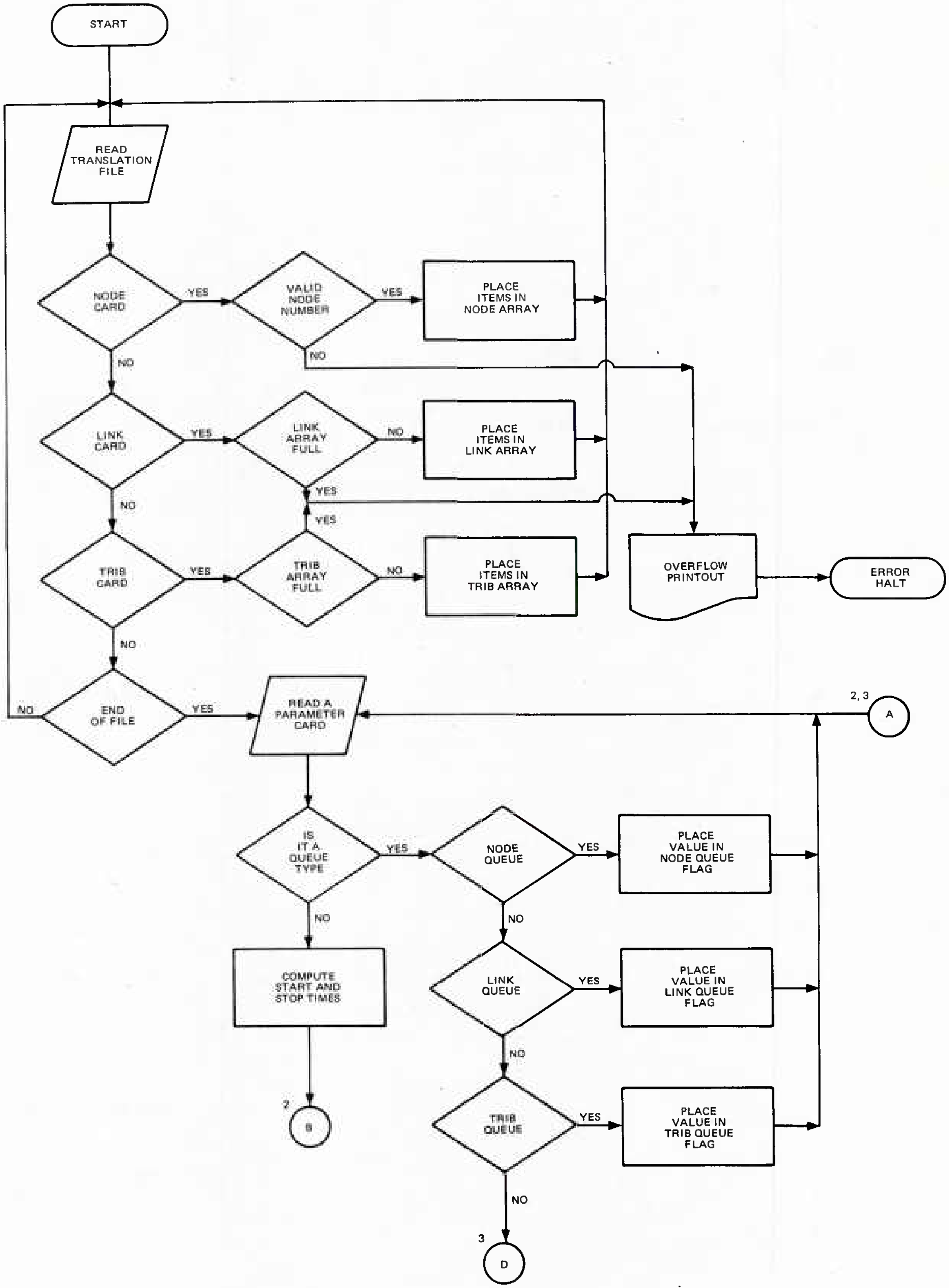
The bulk of the program is written in general FORTRAN IV language and is reasonably machine independent. One small portion of the program uses IBM only "shift" and "logical and" instructions for improved operating speed. These instructions may be replaced by multiply, divide, and subtract instructions for complete independence.

Table 8-1. Parameter Card Format

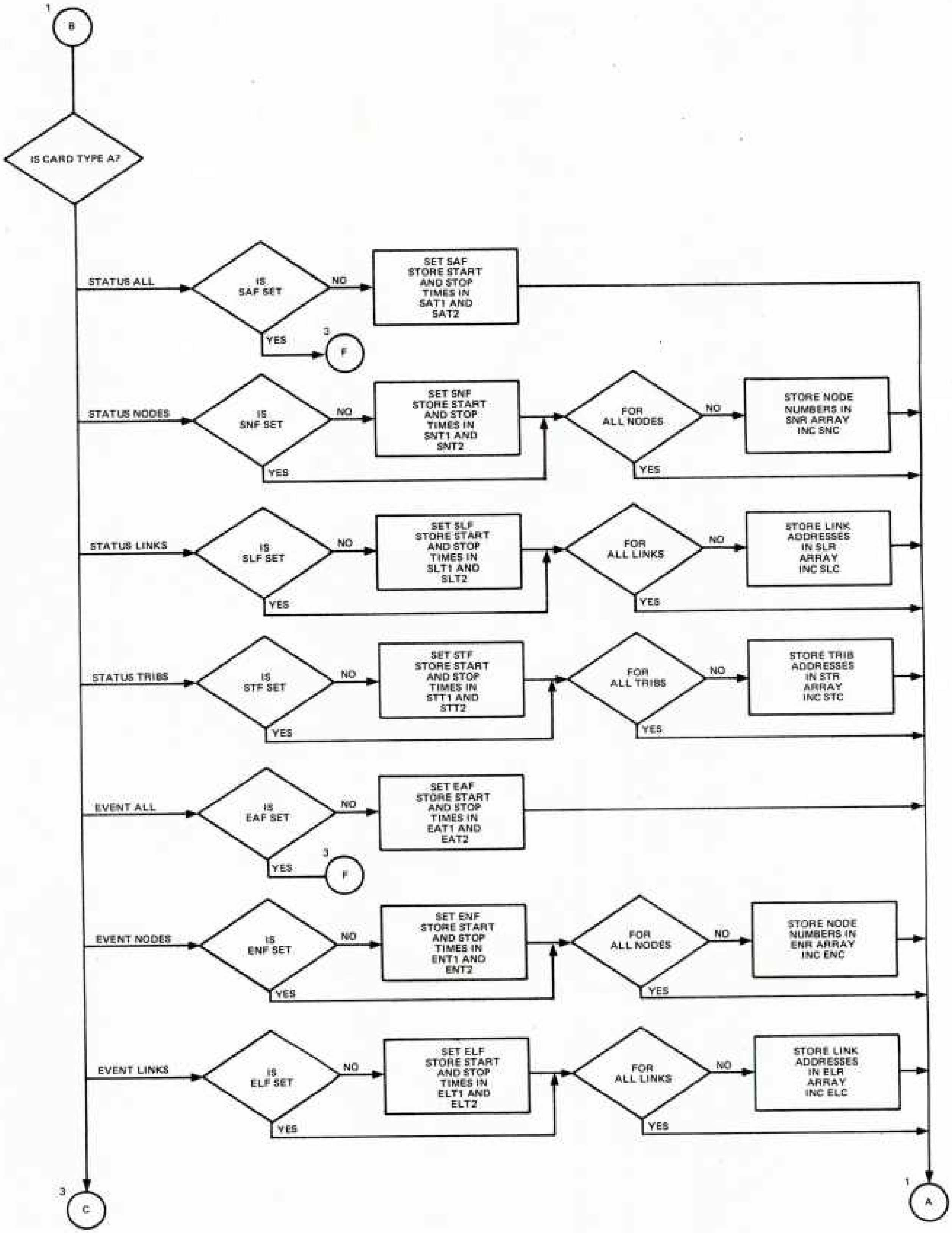
CARD COLUMN	DESCRIPTION	EXPLANATION
1-6	File Type	Type of file to be listed. Only acceptable entries are STATUS or EVENT.
7	Blank	
8-12	Record Type	Defines filtering desired. If file type is STATUS, acceptable entries are ALL, SWT, LINK, TRIB, SWTQ, LINKQ, or TRIBQ. If file type is EVENT, acceptable entries are ALL, SWT, LINK, TRIB, MSG or AC.
13	Blank	
14-18	Queue size	If file type is STATUS and record type is SWTQ, LINKQ, or TRIBQ, this field specifies a queue size. Any of these records whose total queues are equal to, or greater than, this value will be printed. Field is numeric, right justified.
or		
14	Blank	
15-18	START TIME	When not filtering on queue sizes, this is earliest time of a desired record (expressed in military time)
19	Blank	
20-23	STOP TIME	Latest time of a desired record. Valid time range is from 0000 to 3600.
24	Blank	
25-30	Field 1	When filtering on specific switches, links, or tribs is desired, appropriate names are entered in Fields 1 through 8. Entires are left adjusted (with trailing blanks as necessary). When filtering on messages or action codes, entries may be anywhere within the field. If all records of a given type are desired, insert ALL in cc 25-27.
31	Blank	
32-37	Field 2	
38	Blank	
39-44	Field 3	
45	Blank	

Table 8-1. Parameter Card Format (Cont'd)

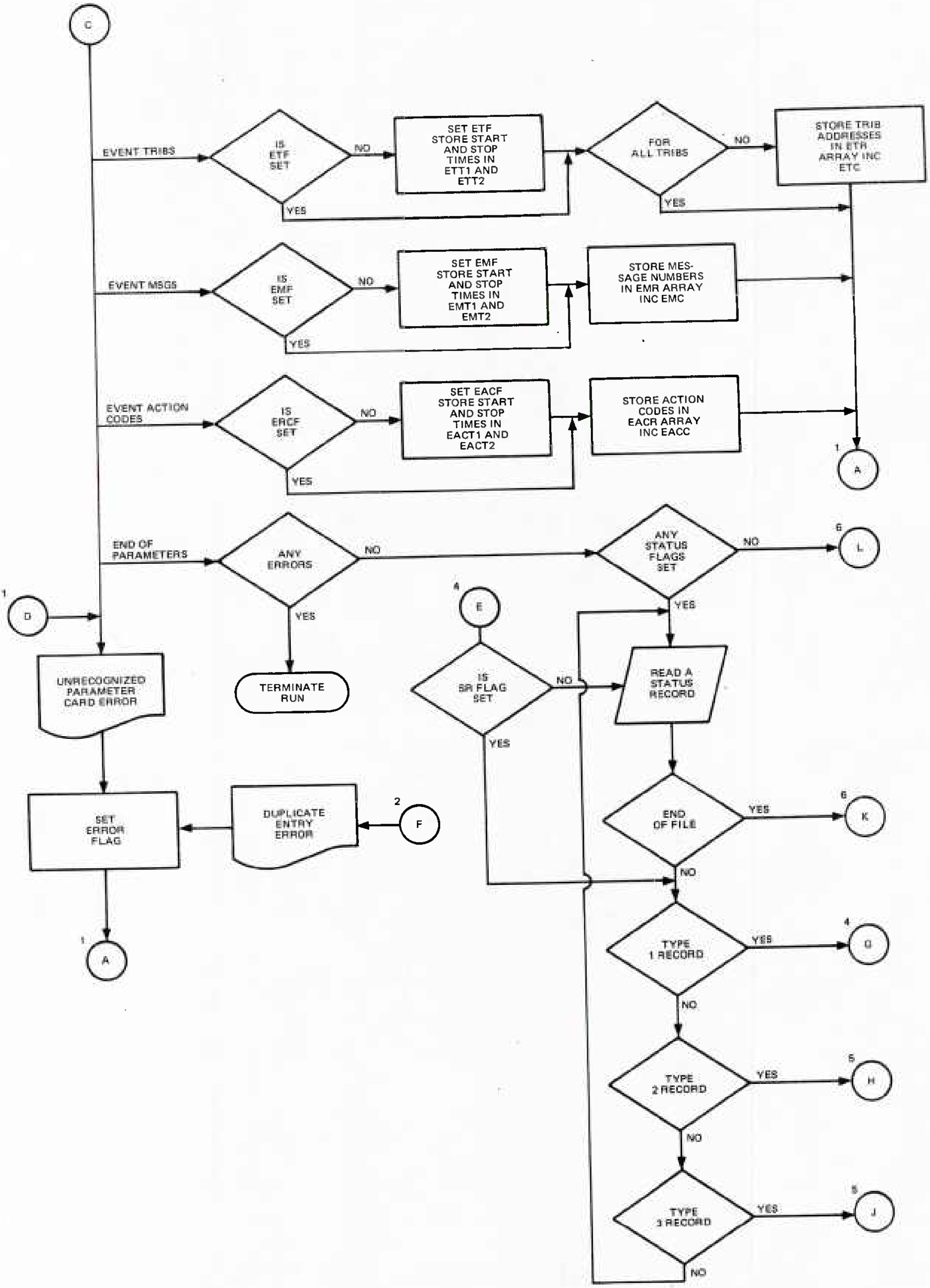
CARD COLUMN	DESCRIPTION	EXPLANATION
46-51	Field 4	
52	Blank	
53-58	Field 5	
59	Blank	
60-65	Field 6	
66	Blank	
67-72	Field 7	
73	Blank	
74-79	Field 8	
80	Blank	



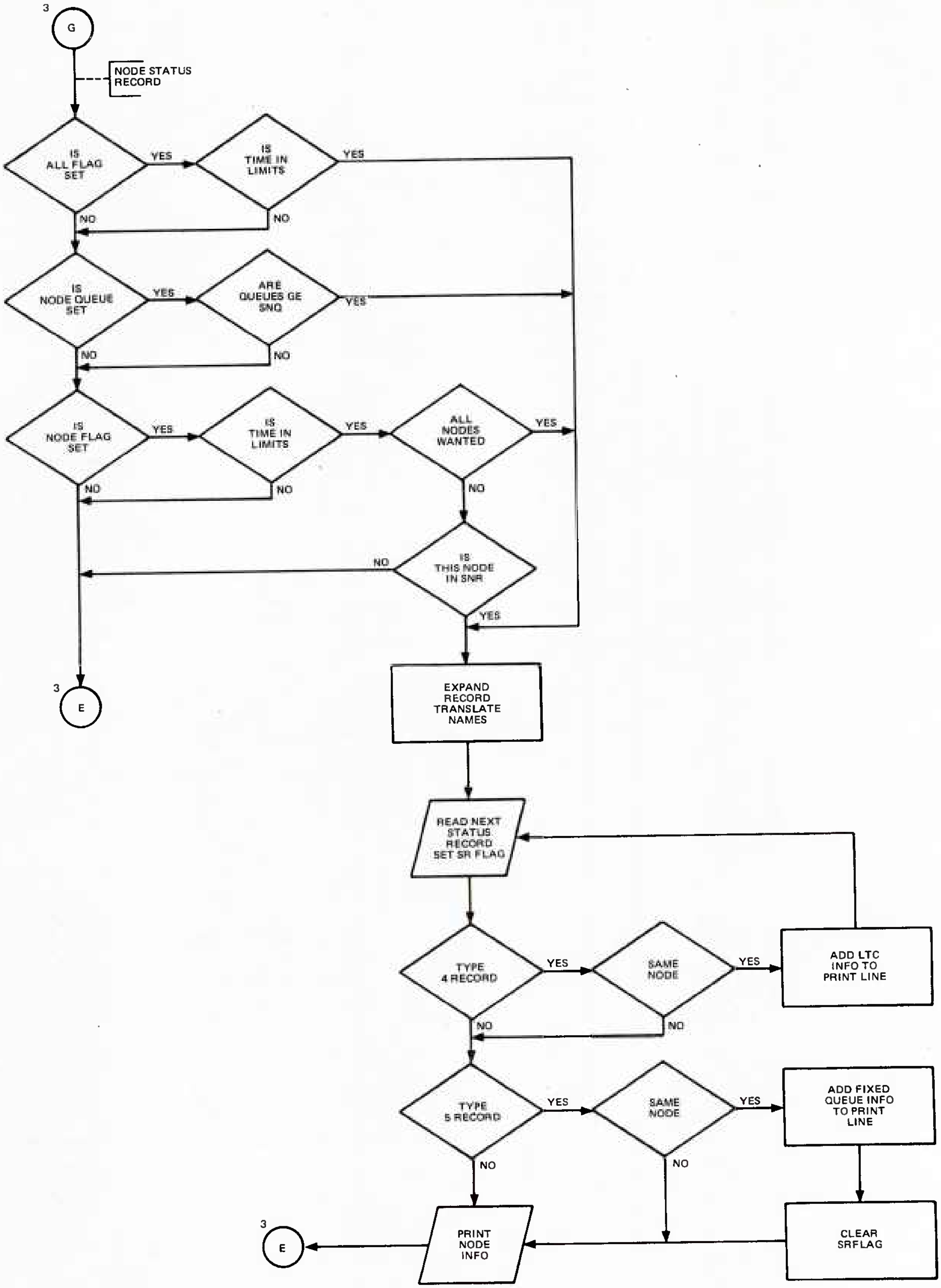
List Events Program (Sheet 1 of 8)


1
B
IS CARD TYPE A?
STATUS ALL
IS SAF SET
NO
YES
3
F
SET SAF STORE START AND STOP TIMES IN SAT1 AND SAT2
STATUS NODES
IS SNF SET
NO
YES
SET SNF STORE START AND STOP TIMES IN SNT1 AND SNT2
FOR ALL NODES
NO
YES
STORE NODE NUMBERS IN SNR ARRAY INC SNC
STATUS LINKS
IS SLF SET
NO
YES
SET SLF STORE START AND STOP TIMES IN SLT1 AND SLT2
FOR ALL LINKS
NO
YES
STORE LINK ADDRESSES IN SLR ARRAY INC SLC
STATUS TRIBS
IS STF SET
NO
YES
SET STF STORE START AND STOP TIMES IN STT1 AND STT2
FOR ALL TRIBS
NO
YES
STORE TRIB ADDRESSES IN STR ARRAY INC STC
EVENT ALL
IS EAF SET
NO
YES
3
F
SET EAF STORE START AND STOP TIMES IN EAT1 AND EAT2
EVENT NODES
IS ENF SET
NO
YES
SET ENF STORE START AND STOP TIMES IN ENT1 AND ENT2
FOR ALL NODES
NO
YES
STORE NODE NUMBERS IN ENR ARRAY INC ENC
EVENT LINKS
IS ELF SET
NO
YES
SET ELF STORE START AND STOP TIMES IN ELT1 AND ELT2
FOR ALL LINKS
NO
YES
STORE LINK ADDRESSES IN ELR ARRAY INC ELC
3
C
1
A

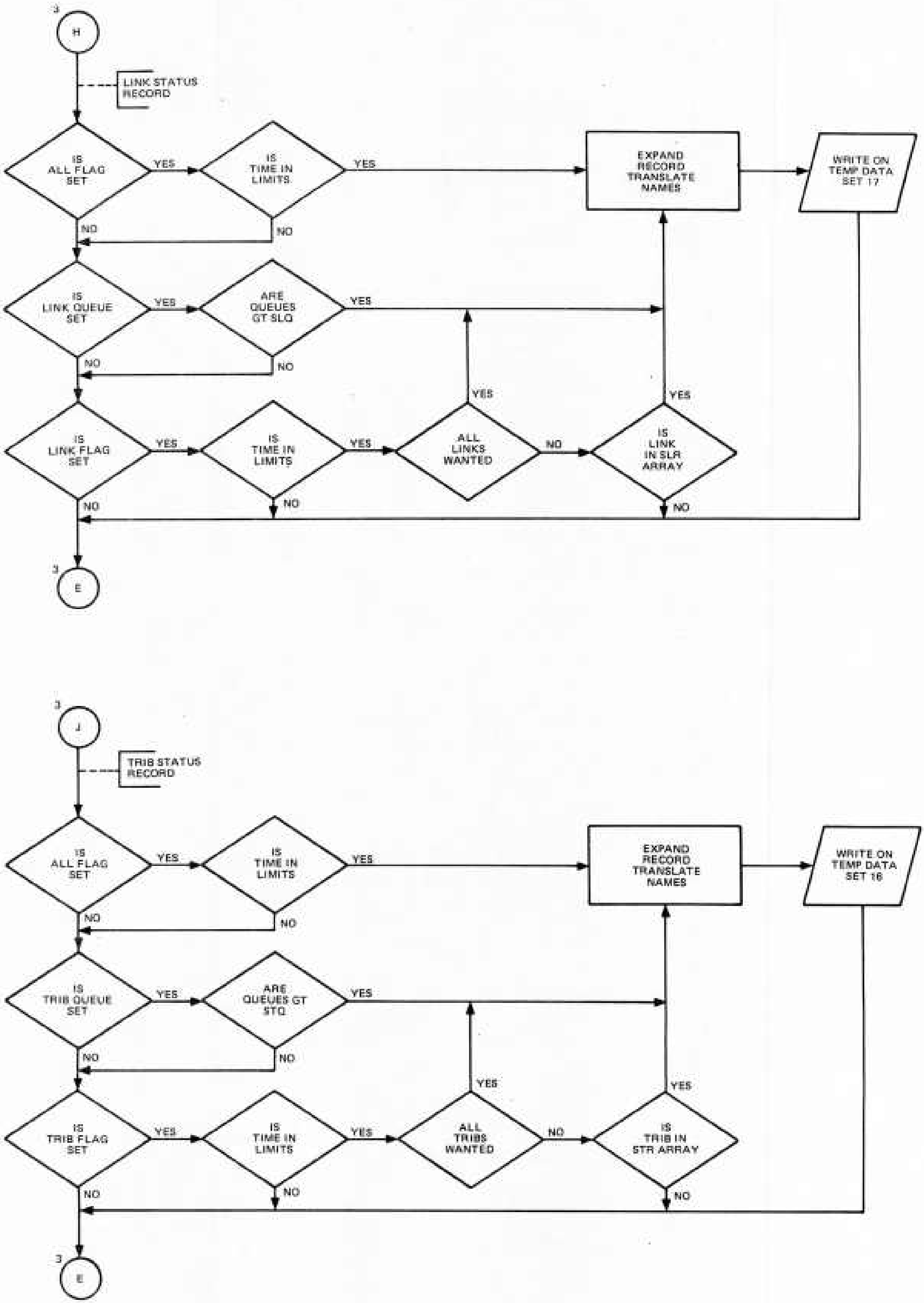
List Events Program (Sheet 2 of 8)



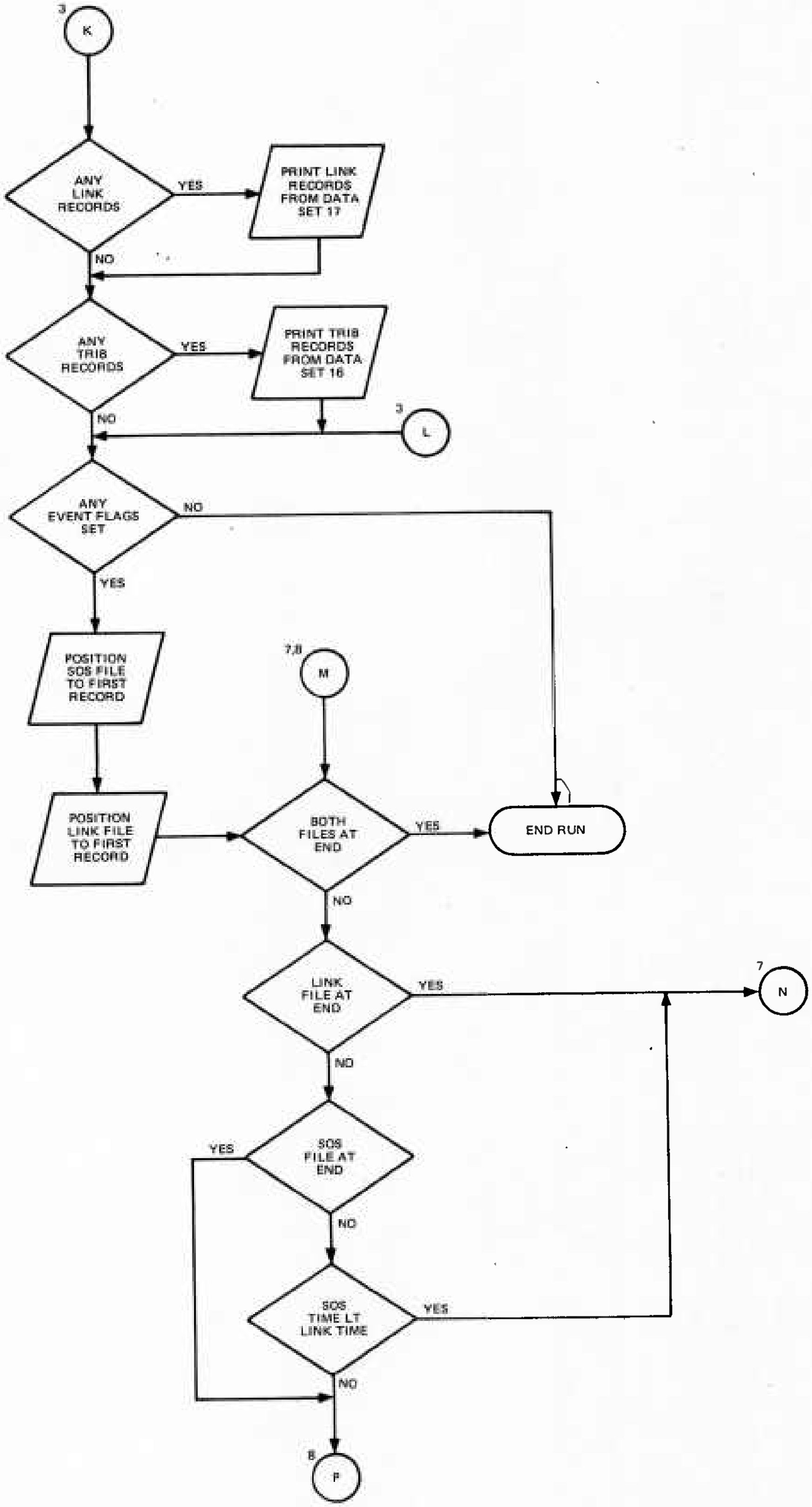
List Events Program (Sheet 3 of 8)



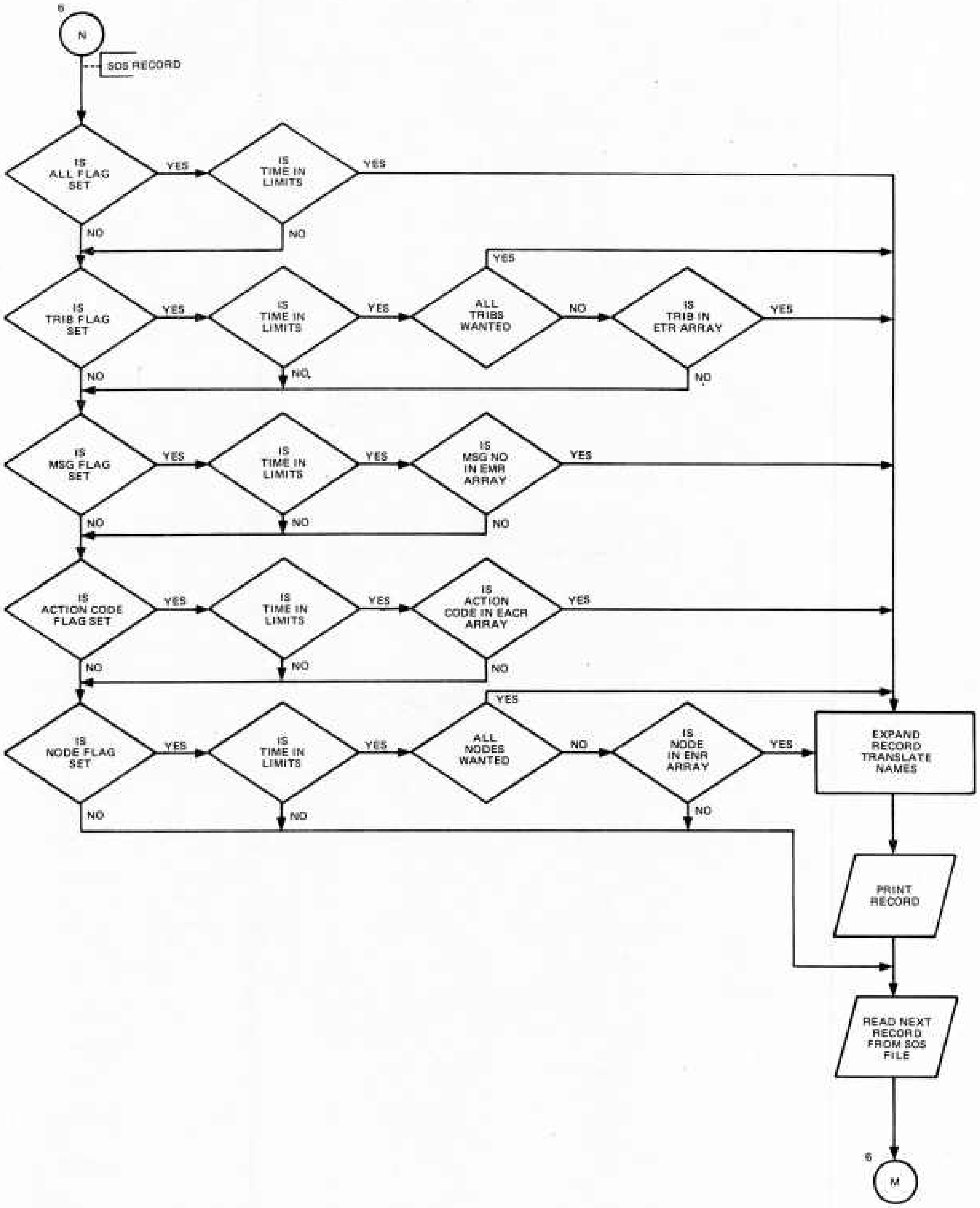
List Events Program (Sheet 4 of 8)



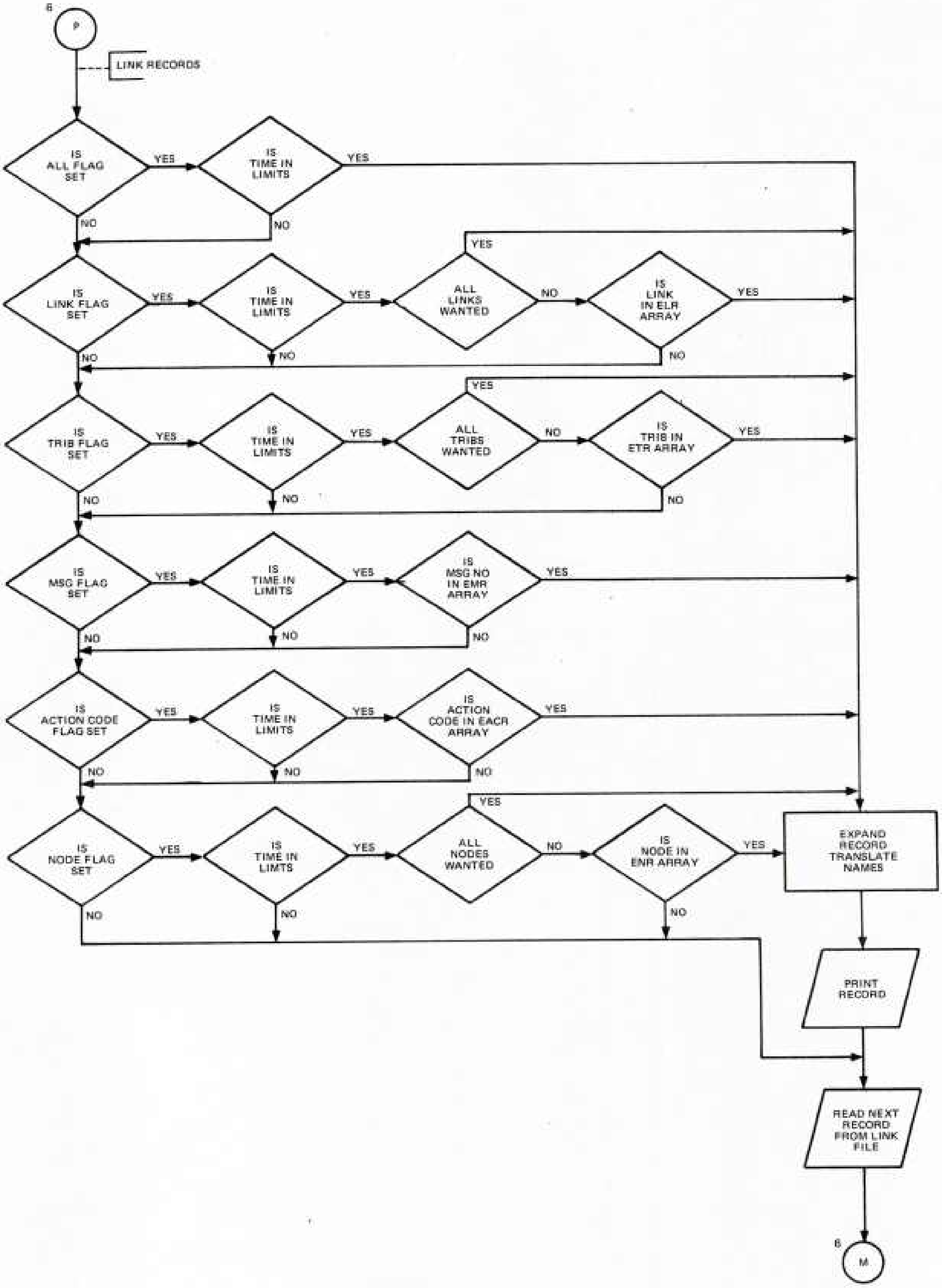
List Events Program (Sheet 5 of 8)



List Events Program (Sheet 6 of 8)



List Events Program (Sheet 7 of 8)



List Events Program (Sheet 8 of 8)

SECTION 9 - HEADER EXTRACT PROGRAM

9.1 PROGRAM REQUIREMENT DESCRIPTION

9.1.1 Program Name: Header Extract Program

9.1.2 Program Identification: DNHDREX1

9.1.3 Purpose

The Header Extract program edits the contents of fields in each message header record on the Header Extract data tapes. Those records, which pass all editing criteria, are used to create the Header Extract Tributary or Header Extract Link files.

9.1.4 Requirements

The Header Extract program is written in ANSI COBOL and requires 74K bytes of core when executed on an IBM 360/65. It requires about 50 minutes of 360/65 CPU time to process a single day's traffic sample of about 700,000 logical records.

9.2 INPUT SPECIFICATIONS

Input to the Header Extract program consists of the Header Extract data tapes. These tapes are organized sequentially with 80-character logical records and 48 logical records per block. About 700,000 records describe a normal day's traffic. The input data is stored on half-inch, seven-track, even-parity magnetic tape with a density of 800 bpi, and coded in BCD. The order in which records are processed is immaterial. Records from a given switching center tend to be contiguous, but may be on separate tapes.

The input records are obtained from the HEXBA1AB and HEXBA1AC tapes created by the Defense Communications Agency (DCA). These Header Extract records are grouped under the name INFILE and contain the following fields are shown in Table 9-1:

1. Precedence - Column 1, precedence, may contain one of six alphabetic precedence characters: R, P, O, Z, W, or Y. R is the lowest precedence, while W (Flash Override) is the highest. Y is the same precedence as W.

2. Language Media Format - Column 2 contains an alphabetic character which indicates the entering and exiting message format.

3. Security - Column 4 contains an alphabetic security character.

4. In-Channel Type - Column 7 contains an alphabetic character which indicates the type and speed of the entering channel.

5. OSRI - Columns 8-14 contain the alphabetic routing indicator of the originating station. The first character of this field is usually R.

6. OSSN - Columns 15-18 contain the message number assigned by the originator.

7. Line Block Count - Columns 19-21 contain the number of line blocks for the message.

8. File Time - Columns 22-28 contain the date-time from the original message header (the file time).

9. System-In Time - Columns 29-35 contain the date-time of the message's original entry into the AUTODIN system.

10. Start of Message In (SOMI) Time - A seven-digit integer in columns 36 through 42 is used to identify the date-time that the message started into the switching center which created the journal tape from which the Header Extract record was derived.

11. End of Message In (EOMI) Time) - Columns 43 through 47 contain a five-digit number which is the date-time of the end of the message corresponding to the SOMI time. The first digit (Column 43) is the last digit of the Julian day and the last four-digits (Columns 44 through 47) are hours and minutes.

12. Start of Message Out (SOMO) Time - Columns 48 through 54 contain a seven-digit representation of the date-time that the message began to leave the switch.

13. End of Message Out (EOMO) Time, Routing Indicator Count, Destination Station Routing Indicator (DSRI) - The routing indicator count, EOMO, and DSRI are interrelated. The DSRI is a seven-character (Columns 62 through 68) alphabetic representation of the final destination of a message. Since messages may be sent to multiple destinations, a message may have more than one DSRI. A separate Header Extract record is created for each DSRI of a message. If a message has multiple destinations, the Header Extract records created for that message are numbered in order of delivery in Columns 60 through 61, the routing indicator count. In all Header Extract records, except the last, pertaining to a given message sent on a given channel, XXXXX appears in the EOMO field (Columns 55 through 59). In the last record, which has the greatest routing indicator count, a time appears as an end of message time in field. The number of transmissions made from the switch is equal to the number of records with an integer time in the EOMO field.

14. Input Channel Number - Each switch numbers its channels somewhat independently. The number of the channel on which the message arrived is in Columns 69 through 71.

15. Destination Numbers - Each routing indicator has associated with it one or more destination numbers which are used to simplify routing. This number is in Columns 72 through 74.

16. Center Designator - Each of the 20 AESCs has associated with it a unique alphabetic character found in Column 75. A list of valid codes is contained in Table 9-3.

17. Destination Channel Number - The number of the channel on which the message was sent is found in Columns 76 through 78.

18. Out Channel Type - Column 79 contains an alphabetic character denoting out channel type.

19. Content Indicator - Column 80 contains a code which is descriptive of the content of the message.

A summary of the Header Extract record appears in Table 9-1. The format used in reading these records is shown in Table 9-2. The symbolic names corresponding to the preceding fields are also summarized in Table 9-2. These names are used throughout the Header Extract program.

9.3 OUTPUT SPECIFICATIONS

The Header Extract Link and Header Extract Tributary files, are created as output by the program. Unacceptable records are printed in an Error Listing with appropriate messages. The program also produces a one-page summary report entitled "AUTODIN Message Volume Summary."

9.3.1 Header Extract Tributary File

The Header Extract Tributary file is organized sequentially on tape with 61-character (EBCDIC) logical records and 70 logical records per block. When the program is run with a full RADAY sample from all primary switches, the Tributary file will contain in excess of 500,000 records on two reels of 1600 BPI magnetic tape.

All records which are identified as containing data representing traffic originating, terminating, or both originating and terminating at a tributary are reformatted and written on the Header Extract Tributary Tape. In addition, the queue time for each message at the AESC (SOMO-EOMI) is computed in minutes and added to each Tributary record. The records on this file are essentially Header Extract records condensed from 80 to 61 characters. The record format is illustrated in Table 9-4.

9.3.2 Header Extract Link File

The Header Extract Link file is sequentially organized on magnetic tape. The logical record length is 29 characters (EBCDIC) with 150 logical records per block. When the program is run with a full RADAY sample from all primary switches, the Link file contains about 200,000 records on one reel of 1600 BPI magnetic tape. All records which are identified as containing data representing traffic passing across interswitch links are reformatted and written on the Header Extract Link Tape. The queue time is computed for each record and added to each link record. The records on this file are essentially Header Extract records condensed from 80 to 29 characters. The record format is illustrated in Table 9-5.

9.3.3 AUTODIN Message Volume Summary

The AUTODIN Message Volume Summary, a one-page summary report, presents, in tabular form, the total message volume by major AUTODIN switch for each of up to 14 unique RADAYs. Message counts which appear in all RADAY columns, other than for the sample RADAY, represent records in which the time-of-transmission (Day Field) occurred on a day other than the sample RADAY. This can be the result of traffic which entered the system prior to the sample RADAY, but which also terminated or passed through a major switch on the sample RADAY. This can also occur as the result of substitution of one or more Header Extract switch samples from a RADAY other than the specified RADAY.

9.4 PROGRAM LOGIC

9.4.1 Program Execution

The program consists of two main segments. The first reads a record from the input file, edits the record, and writes it out on the appropriate output file. If the record is found to be in error, it is listed with an appropriate message on the Error Listing. Edit criteria, data conversion rules, error messages, and program error reaction procedures are listed in Table 9-1. If the record is found to be acceptable, the internal message count table (PRI-ARRAY-TAB) is incremented by one for the

RADAY specified in the record. The message counts for each switch are broken down for each of up to 14 unique RADAYs, with message counts for all RADAYs in excess of 14 grouped under a RADAY OTHER column.

The second segment computes the PRI-ARRAY-TAB row and column totals. The AUTODIN Message Volume Summary is then produced using the PRI-ARRAY-TAB count and computed totals and subtotals.

9.4.2 Internal Primary Switch Table Modification

The one-page AUTODIN Message Volume Summary presents message subtotals for CONUS, PAC, and EUR. The program identifies the switches in each geographic region through positional location in the internal SWITCH-CONVER-TAB. When changes occur to the AUTODIN primary switch designators, such as when switches are added to or deleted from AUTODIN, this table must be modified. Table 9-6 illustrates the current table layout and provides instructions for table modification.

The program logic flow charts are included at the end of this section.

9.4.3 Memory Layout

The program does not use subroutines and requires a single block of memory for executable instructions and storage arrays.

9.5 MACHINE DEPENDENCE

In general, the Header Extract program is machine independent. All coding conforms to ANSI standards. However, in implementing the program on an IBM 360/65. some machine dependence was unavoidable. The Identification Division describes an IBM 360/65.

Table 9-1. Header Extract Input Record Format and Edit Criteria

COLUMN	FIELD	ACCEPTABLE CHARACTERS	ERROR LABEL OR ACTION
1	Message Precedence	Z, O, P, W, R; Y changed to Z	Value changed to R
2	Language Media Format	A, F, Q, R, T for Teletype changed to T; or B, D, I for Magnetic Tape changed to M; or C, S for Punched Card changed to C	LMF ERROR
3	Language Media Format (Not Used)		
4	Message Security	A, T, S, C, U; E changed to U; R changed to C	Value changed to U
5-6	Blank		
7	Input Channel Type	A, B, C, D, E, F, G, H, K, M, S, T, U, V, Z, * and Blank	INCHATYP ERROR
8-14	Originating Station Routing Indicator	First character must equal R	OSRI ERROR
15-18	Originating Station Serial Number	All characters must be numeric	OSSN ERROR
19-21	Line Block Count	001 - 550; changed to 015 if LMF=T and LBC=550; 030 if LMF=C and LBC=550	LBC ERROR
22-28	File Time From Message		
29-31	Time of Transmission: Day	001 - 366	TOT ERROR
32-35	Time of Transmission: Hour	0000 - 2400	TOT ERROR
	The following eight fields represent time into or out of the reporting AESC:		
36-38	Start of Message IN: Day	001 - 366	SOMI ERROR
39-42	Start of Message IN: Hour	0000 - 2400	SOMI ERROR
43	End of Message IN: Day	0 - 9	EOMI ERROR
44-47	End of Message IN: Hour	0000 - 2400	EOMI ERROR
48-50	Start of Message OUT: Day	001 - 366	SOMO ERROR
51-54	Start of Message OUT: Hour	0000 - 2400	SOMO ERROR
55	End of Message OUT: Day	0 - 9 or X	EOMO ERROR
56-59	End of Message OUT: Hour	0000 - 2400 or XXXX	EOMO ERROR
60-61	Routing Indicator Count (Not Used)		
62-68	Destination Routing Indicator	First Character must equal R	DSRI ERROR
69-71	Input Channel Number	SVM, ICI, IMI, or Numeric	INCHANR ERROR
72-74	Destination Number (Not Used)		
75	Code of Reporting AESC	A, B, C, D, E, F, G, H, J, K, M, N, P, Q, R, S, T, U, V, W, X, Y	CENDEG ERROR
76-78	Output Channel Number	ICO, IMO, or Numeric	OCHANR ERROR
79	Output Channel Type	A, B, C, D, E, F, G, H, K, M, S, T, U, V, Z, * and Blank	OCHATYP ERROR
80	Content Indicator (Not Used)		

Table 9-2. COBOL Header Extract Record Format

MNEMONIC REFERENCE	LEVEL	NUMBER OF CHARACTERS	ITEM DESCRIPTION
INFILE	FILE	80	Header Extract File
DATAIN	RECORD	80	
PRECED	FIELD	1	Precedence
LMF	FIELD	1	Language Media Format
FILLER	FIELD	1	
SECUR	FIELD	1	Security
FILLER	FIELD	2	
INCHATP	FIELD	1	Input Channel Type
OSRI	GROUP	7	Origin Routing Indicator
OSRI-A	FIELD	1	
OSRI-B	FIELD	6	
OSSN	FIELD	4	Origin Serial Number
LBC	FIELD	3	Line Block Count
FILLER	FIELD	3	
FILLER	FIELD	4	
TOT	GROUP	7	Time of Transmission (TOT)
TOT-D	FIELD	3	TOT: (RADAY)
TOT-H	FIELD	4	TOT: Hour
SOMI	GROUP	7	Start of Message In
SOMI-D	FIELD	3	SOMI: Day
SOMI-H	FIELD	4	SOMI: Hour
EOMI	GROUP	5	End of Message In
EOMI-D	FIELD	1	EOMI: Day
EOMI-H	FIELD	4	EOMI: Hour
SOMO	GROUP	7	Start of Message Out
SOMO-D	FIELD	3	SOMO: Day
SOMO-H	FIELD	4	SOMO: Hour
EOMO	GROUP	5	End of Message Out
EOMO-D	FIELD	1	EOMO: Day
EOMO-H	FIELD	4	EOMO: Hour
FILLER	FIELD	2	
DSRI	FIELD	7	Destination Routing Indicator
INCHANR	FIELD	3	Input Channel Number
FILLER	FIELD	3	
CENDEG	FIELD	1	Reporting Switch Code
OCHANR	FIELD	3	Output Channel Number
OCHNTYP	FIELD	1	Output Channel Type
FILLER	FIELD	1	

Table 9-3. AUTODIN Node Designation

ROUTING INDICATOR	SWITCH DESIGNATOR	NAME
RUAD	J	Camp Drake
RUAO	X	Fort Buckner
RUCI	G	Gentile
RUCL	B	Albany
RUDO	E	Croughton
RUEB	A	Andrews
RUED	H	Hancock
RUEO	D	Ft. Detrick
RUFL	Q	Coltano
RUFT	P	Pirmasens
RUHH	W	Wahiawa
RUHJ	F	Guam
RUMM	C	Clark
RUMO	K	Konat
RUMU	S	Phu Lam
RUWJ	N	Norton
RUWM	M	McClellan
RUWT	T	Tinker
RUAK	R	Korea
RUFF	Y	Germany

Table 9-4. COBOL Header Extract Tributary File Record Format

MNEMONIC REFERENCE	LEVEL	NUMBER OF CHARACTERS	ITEM DESCRIPTION
HEXTRIB	FILE	61	Header Extract Trib File
OUTREC-TRIB	RECORD	61	
RADAY	FIELD	3	Time of Transmission: Day
PRECED	FIELD	1	Precedence
SECUR	FIELD	1	Security
LMF	FIELD	1	Language Media Format
CENDEG	FIELD	1	Reporting Switch Code
OSRI	FIELD	7	Origin Routing Indicator
INCHANR	FIELD	3	Input Channel Number
OCHANR	FIELD	3	Output Channel Number
INCHATP	FIELD	1	Input Channel Type
OCHNTYP	FIELD	1	Output Channel Type
OSSN	FIELD	4	Origin Serial Number
LBC	FIELD	3	Line Block Count
TOT-H	FIELD	4	Time of Transmission: (in minutes)
DSRI	FIELD	7	Destination Routing Indicator
EOMO-D	FIELD	1	End of Message Out: Day
EOMO-H	FIELD	4	End of Message Out: Hour
SOMO	FIELD	7	Start of Message Out
EOMI	FIELD	5	End of Message In
IN-Q-T	FIELD	4	Time Message in Queue in Minutes

Table 9-5. COBOL Header Extract Link File Record Format

MNEMONIC REFERENCE	LEVEL	NUMBER OF CHARACTERS	ITEM DESCRIPTION
HEXLINK	FILE	29	Header Extract Link File
OUTREC-LINK	RECORD	29	
SW-NM-T	FIELD	4	Reporting Switch Name
PRECED	FIELD	1	Precedence
OCHANR	FIELD	3	Output Channel Number
SOMI	FIELD	7	Start of Message In
LBC	FIELD	3	Line Block Count
SOMO	FIELD	7	Start of Message Out
IN-Q-T	FIELD	4	Time Message in Queue in Minutes

Table 9-6. Internal Switch Table Sequence

PAC	CONUS	EUR
1 RUAD	8 RUCI	17 RUDO
2 RUAO	9 RUCL	18 RUFL
3 RUHJ	10 RUEB	19 RUFT
4 RUMM	11 RUED	20 RUFF
5 RUMO	12 RUEO	
6 RUMU	13 RUHH	
7 RUAK	14 RUWJ	
	15 RUWM	
	16 RUWT	

If changes are made:

1. PAC-CONUS-EUR sequence must be maintained
2. Moves to initialize the table will probably have to be reordered to maintain table continuity.
3. The following fields must be changed to reflect proper groupings and length:

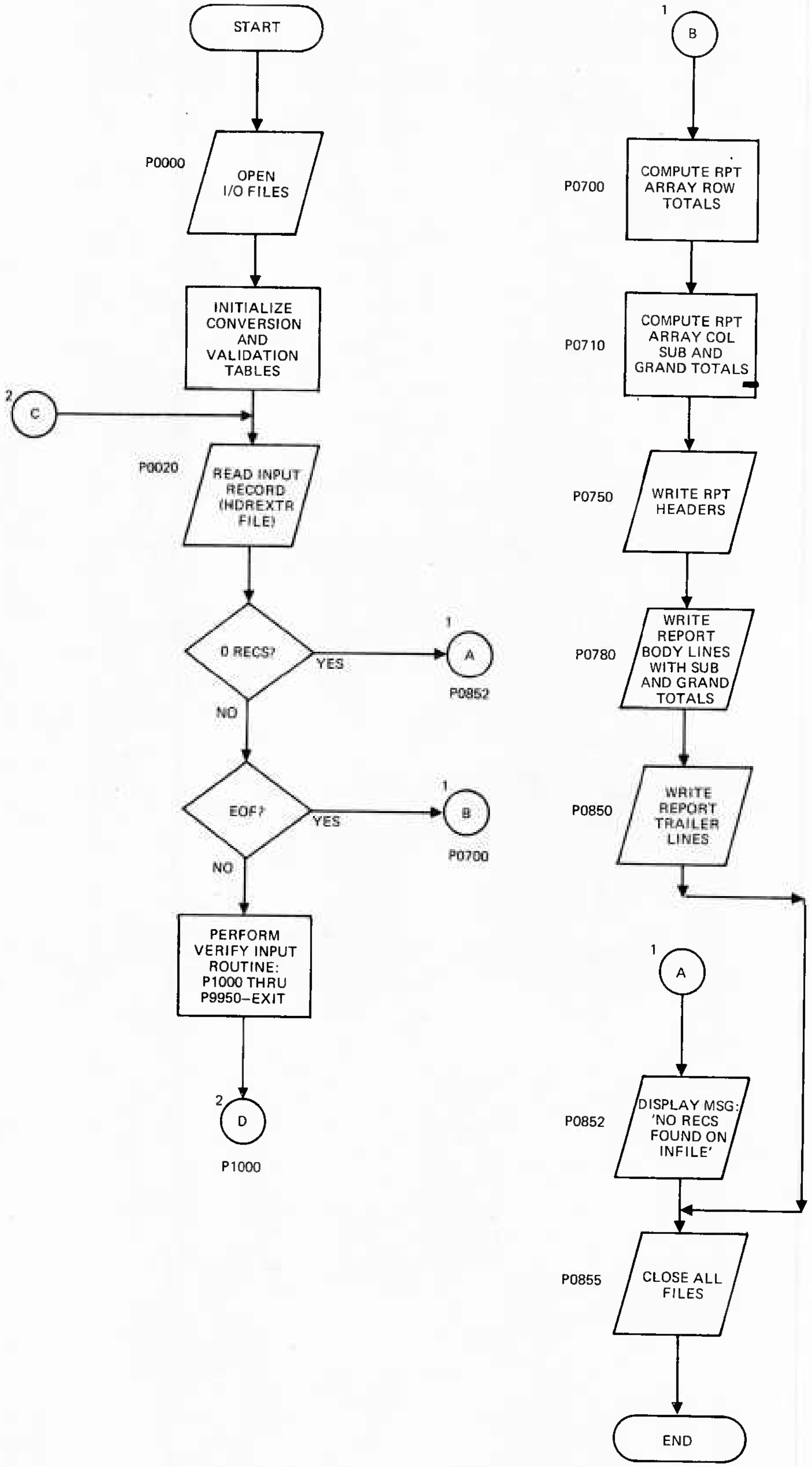
   (a) SW-COUNT-PLUS1 *(=21)

   (b) LAST-PAC-POS-IN-SWTAB-PLUS1 (=8)

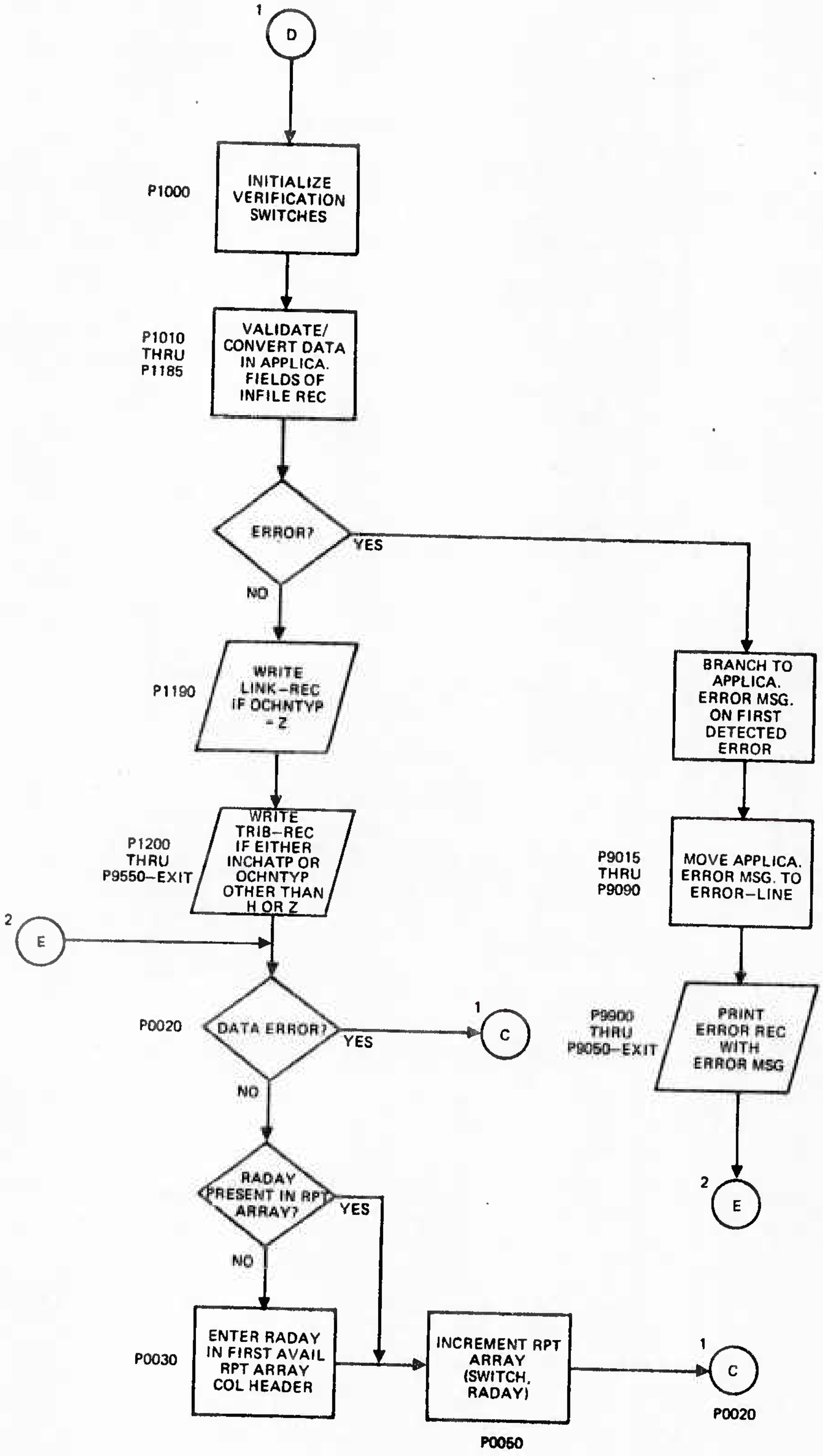
   (c) LAST-CONUS-POS-IN-SWTAB-PLUS1 (=17)

   (d) LAST-EUR-POS-IN-SWTAB-PLUS1 (=21)

* [(=nn) : current values]


START
P0000
OPEN I/O FILES
INITIALIZE CONVERSION AND VALIDATION TABLES
2
C
P0020
READ INPUT RECORD (HDREXTR FILE)
0 RECS?
YES
1
A
P0852
NO
EOF?
YES
1
B
P0700
NO
PERFORM VERIFY INPUT ROUTINE: P1000 THRU P9950–EXIT
2
D
P1000
1
B
P0700
COMPUTE RPT ARRAY ROW TOTALS
P0710
COMPUTE RPT ARRAY COL SUB AND GRAND TOTALS
P0750
WRITE RPT HEADERS
P0780
WRITE REPORT BODY LINES WITH SUB AND GRAND TOTALS
P0850
WRITE REPORT TRAILER LINES
1
A
P0852
DISPLAY MSG: 'NO RECS FOUND ON INFILE'
P0855
CLOSE ALL FILES
END

Header Extract Program (Sheet 1 of 2)


1
D
P1000
INITIALIZE VERIFICATION SWITCHES
P1010 THRU P1185
VALIDATE/ CONVERT DATA IN APPLICA. FIELDS OF INFILE REC
ERROR?
YES
NO
P1190
WRITE LINK-REC IF OCHNTYP = Z
BRANCH TO APPLICA. ERROR MSG. ON FIRST DETECTED ERROR
P1200 THRU P9550-EXIT
WRITE TRIB-REC IF EITHER INCHATP OR OCHNTYP OTHER THAN H OR Z
P9015 THRU P9090
MOVE APPLICA. ERROR MSG. TO ERROR-LINE
2
E
P0020
DATA ERROR?
YES
1
C
P9900 THRU P9050-EXIT
PRINT ERROR REC WITH ERROR MSG
NO
RADAY PRESENT IN RPT ARRAY?
YES
2
E
NO
P0030
ENTER RADAY IN FIRST AVAIL RPT ARRAY COL HEADER
INCREMENT RPT ARRAY (SWITCH, RADAY)
P0050
1
C
P0020

Header Extract Program (Sheet 2 of 2)

SECTION 10 - AMIE EXTRACT PROGRAM

10.1 PROGRAM REQUIREMENTS DESCRIPTION

10.1.1 Program Name: AMIE Extract Program

10.1.2 Program Identification: DNAMIEX1

10.1.3 Purpose

The AMIE Extract program edits the contents of fields in each record on the AUTODIN Management Index (AMIE) file tape. Those records which pass all editing criteria and represent AUTODIN tributaries or links are used to create the AMIE Extract Tributary and AMIE Extract Link files.

10.1.4 Requirements

The AMIE Extract program is written in ANSI COBOL and requires 94K bytes of core when executed on an IBM 360/65.

10.2 INPUT SPECIFICATIONS

There is one input to the program; the DCS AUTODIN Management Index (AMIE) file.

10.2.1 File Characteristics

The AMIE Extract tape is organized sequentially with the first logical record on the tape containing 80 characters of file header information. All other logical reocrds on the tape contain 128 characters. The file is unblocked and typically, there are about 1500 records selected by the program. The input data is stored on half-inch, 800 bpi, BCD, seven-track, even-parity magnetic tape. The order in which records are processed is immaterial.

10.2.2 Record Description

The input records are obtained from the AMIE file, which is created and maintained by the Defense Communications Agency within an IBM 1410 computer environment.

The first record on the AMIE file is a header record as illustrated in Table 10-1. The AMIE file data record layout, with permissible field values, is also illustrated in Table 10-1. A description of the fields in the data record are:

1. Circuit ID Number - Columns 1-5 in each record contain the subscriber ID number or commercial circuit number. Those records containing the value 9 in the first position of this field (Column 1) describe the AUTODIN links.

2. Center Designator - The switching center to which the circuit described in the record is assigned is identified by an alphabetic character in Column 6. A list of valid switch codes is contained in Table 9-3.

3. Routing Indicator Type - Each terminal has a prime routing indicator and can have a number of other routing indicators such as PSEUDO, DATA, or CONTINGENCY. Only those records containing a primary routing indicator code A in Column 7 are selected by the program.

4. Routing Indicator - Columns 8-14 contain the alphabetic routing indicator of the tributary or link. If the record describes a link, the value must contain a valid, left justified, four-character switch identification. This identification represents, in the case of a link record, the origin switch. The routing indicator for a tributary is not obtained from this field, but is obtained from the secondary routing indicator (Columns 74-80).

5. Circuit Speed Code - Columns 59-60 contain the circuit speed code. Table 10-2 lists the baud rate for each code.

6. Circuit Security - The code for the maximum security of the circuit is in Column 64.

7. Channel Number - The channel number assigned to the tributary or link is in Columns 67-69.

8. Secondary Routing Indicator Type Code - For records describing links, the value A in Column 72 identifies the secondary routing indicator (Columns 74-80) as a prime routing indicator; i. e. the destination switch of a link. Each

tributary has a primary routing indicator, but can also have a number of other secondary routing indicators. The secondary routing indicator type codes A (Primary), D (Data), and P (Pseudo), identify the tributary secondary routing indicator appearing in Columns 74-80. Only tributary records containing these codes are of interest.

9. Secondary Routing Indicator - For records describing links and containing a secondary routing indicator type code of A, the value found in Columns 74-80 is the destination switch of a link and must be a primary switching center. For tributary records containing a secondary routing indicator type code of A, D, or P, this field contains the secondary routing indicator for the tributary. It is this value that is used to identify each tributary placed in the AMIE Tributary file.

10. Line Traffic Controller (LTC) Number - Column 81 contains a numeric value of 1, 2, 3, or 4, which identifies the LTC to which the channel is attached.

11. Date Code - Column 86 contains a code which indicates the status of the link or tributary. Codes B (Pending Test Circuit), G (Contingency Circuit). and P (Future Circuit), represent records which are of no interest in that they are not normally active.

A summary of the AMIE Extract record appears in Table 10-1. The format used in reading these records is shown in Table 10-3. The symbolic names are also summarized in Table 10-3.

10.3 OUTPUT SPECIFICATIONS

Two disk resident files, the AMIE Extrack Link and AMIE Extract Tributary files are created as output by the program. In addition, three listings of these files are produced by the program. Unacceptable records are printed in an error listing together with appropriate messages.

10.3.1 AMIE Link File

All AMIE Extract records which are identified by the program as containing data defining a one-way path between two switching centers are reformatted and written on the AMIE Link file in card image format. The AMIE Link file is organized sequentially on disk with 80-character (EBCDIC) logical records and 100 records per block. The record format is illustrated in Table 10-4.

10.3.2 AMIE Tributary File

All AMIE Extract records, which are identified by the program as containing data defining an active terminal station directly connected to the AUTODIN system, are reformatted and written on the AMIE Tributary file in card image format. The AMIE Tributary file is organized sequentially on disk with 80-character (EBCDIC) logical records and 100 records per block. The record format is illustrated in Table 10-t.

10.3.3 AMIE Link File Listing

The AMIE Link file listing is presented in source switch, destination switch, and channel number sort sequence. The LTC number and circuit speed (in milliseconds per line block) are displayed for each link entry. Multiple entries may appear in this listing where more than one circuit speed is possible for the link. This listing displays all multiple speed entries for a link and identifies those entries which are actually written in the AMIE Link file. Only one recrod for each link is placed in this file.

10.3.4 AMIE Tributary File Listings

The program produces two output listings of the AMIE Tributary file in different sort orders. The first presents the tributaries in source switch and channel number sequence. Up to nine secondary routing indicators may appear next to each terminal listed. The second presents the tributaries sorted by secondary routing indicator, source switch, and channel number. This report also presents the LTC number, the equipment code number, and the circuit speed (in milliseconds per line block) for each entry.

10.4 PROGRAM LOGIC

10.4.1 Program Execution

The program consists of four main segments. The first reads the AMIE Extract Tape header record, validates its contents, and converts the Julian date to Gregorian for report identification.

The remaining three segments consist of a COBOL Sort Input Procedure, Sort, and COBOL Sort Output Procedure. The logic contained in the Sort Input Procedure selects applicable link and tributary records from the AMIE Input file, edits the field contents of these records, and releases them to the Sort file with the necessary sort key values. Each tributary record is released to the sort twice with primary key values of A and B. A and B records contain different tributary data as illustrated in Table 10-6. Link records are released to the Sort with a primary key value of C and with data as shown in Table 10-6. After the last AMIE input file record is processed, the Sort file is sorted in the sequence specified in Table 10-6.

Execution of the Sort Output Procedure Logic is controlled by the primary Sort key value of A, B, or C. "A" records are used to produce the Tributary listing containing all secondary routing indicators assigned to a given tributary. "B" records are used to produce the Tributary listing sorted by secondary routing indicator, and to produce the Tributary file. "C" records are used to generate both the Link file and Link listings.

10.4.2 Internal Primary Switch Table Modification

The program contains an internal table (SWITCH-CONVER-TAB) which contains a list of all valid AUTODIN primary switches and their respective one-character codes. The table is capable of containing 24 switch values, but is currently initalized with 21 switches. If the table is modified, the field SW-COUNT-PLUS1 must be set to the value in the table plus one. Switches may appear in any order within the table.

10.4.3 Error Diagnostics

1. OUTPUT FROM SORT CONTAINS 0 RECORDS

A legal header record was found on the AMIE input file, but no valid tributary or link records were detected by the program. The Error Listing should be examined and if this provides insufficient explanation for the error, a dump of the AMIE input tape should be obtained for examination.

2. SORT CONTAINS A AND B RECORDS ONLY

One or more legal tributary records were found on the AMIE input file, but no valid link records were detected by the program. The Error Listing should be examined and if this provides insufficient explanation for the error, a dump of the AMIE input tape should be obtained for examination.

3. FST REC IN SORT CONTAINS KEY VAL OF C

One or more legal link records were found on the AMIE input file, but no valid tributary records were detected by the program. See (2) for corrective action.

4. FST REC IN SORT CONTAINS KEY VAL OF B
FST REC IN SORT CONTAINS KEY VAL OF C
FST REC IN SORT CONTAINS INVALID KEY VAL
FTS REC IN SORT GP 2 CONTAINS INVALID KEY
SORT CONTAINS A RECORDS ONLY
FST REC IN SORT GP 3 CONTAINS INVALID KEY
ADDITIONAL RECORDS FOLLOW C GROUP IN SORT

An internal program error has occurred during the generation or processing of the Internal Sort file.

Program logic flow charts are included at the end of this section.

10.4.3 Memory Layout

The program does not use subroutines and requires a single block of memory for executable instructions and storage arrays.

10.5 MACHINE DEPENDENCE

In general, the AMIE Extract program is machine independent. All coding conforms to ANSI standards. However, in implementing the program on an IBM 360/65, some machine dependence was unavoidable. The IDENTIFICATION DIVISION describes an IBM 360/65.

Table 10-1. AMIE Extract Record Format and Edit Criteria

COLUMN	FIELD	ACCEPTABLE CHARACTERS	ERROR LABEL OR ACTION
1-4	Constant	"1HDR"	HDR CONST A NOT EQUAL TO 1HDR
5-19	Blank		
20-29	Constant	"AMIE DUMP "	HDR CONST B NOT EQUAL TO AMIE DUMP
30-31	Julian Date-Year	56-96	HDR DATE IN ERROR
32-34	Julian Date-Day	001-366	HDR DATE IN ERROR
35-80	Blank		
DATA RECORD			
1-5	ID Number or Commercial Communications Number ID Number C1	Link Rec: 9 Trib Rec: No check	Record passed to Trib Rec Logic
6	Switch Code	A, B, C, D, E, F, G, H, J, K, M, N, P, Q, R, S, T, W, X, Y	ERROR - INVALID ASC
7	Routing Indicator Type	Link Rec: A Trib Rec: A	Bypass Record Processing Bypass Record Processing
		Not Equal Blank	ERROR - RI-TYP BLANK
8-14	Routing Indicator	Link Rec: Valid ASC	ERROR - RI-INVALID
15	DCA Area Code		
16-19	Operating Command Code		
20-33	Activity		
34-41	Geographical Location		
42-43	State/Country Code		
44	Card Number		
45-52	Communication Circuit System Designator		
53-58	Terminal Equipment Code		
59-60	Circuit Speed Code	AM, AH, AL, AP, AR, AT, AN, AC	ERROR - INVALID SPEED
61-63	Message Format		
64	Security		
65	Hours of Operation Code		
66	Channel Type Code		
67-69	Channel Number	Not Equal Blank	ERROR - CHN BLANK
70-72	Destination Number		
73	Secondary Routing Indicator Type Code	Link Rec: A Trib Rec: A, D, P	Bypass Record Processing Bypass Record Processing
		Not Equal Blank	ERROR - SECRI TYP BLANK
74-80	Secondary Routing Indicator	Trib Rec: C1 = R	ERROR - SECRI-C1 NE R
81	Line Traffic Controller	1, 2, 3, 4	Move 1 to the LTC
82-85	Blank		
86	Date Code	Not Equal P, B, G	Bypass Record Processing
87-128	Blank		

Table 10-2. Circuit Speed Codes

CODE	BAUD RATE
AC	45
AH	75
AL	150
AM	300
AN	600
AP	1200
AR	2400
AT	4800

Table 10-3. AMIE Extract Record Format

MNEMONIC REFERENCE	LEVEL	NUMBER OF CHARACTERS	ITEM DESCRIPTION
INFILE	FILE	80 to 128	AUTODIN Management Index File
HDRIN	RECORD	80	AMIE File Header Record
HDR-CONST-A	FIELD	4	VALUE = 1HDR
FILLER	FIELD	15	
HDR-CONST-B	FIELD	10	VALUE = AMIE DUMP
HDR-DATE	GROUP	5	File Julian Date
HDR-YEAR	FIELD	2	Date: Year
HDR-DAY	FIELD	3	Date: Day
FILLER	FIELD	46	
DATAIN	RECORD	128	AMIE File Data Record
ID-NR	GROUP	5	Subscriber ID or Commercial Circuit Number
ID-NR-A	FIELD	1	
ID-NR-B	FIELD	4	
SW-CD	FIELD	1	Reporting Switch Code
RI-TYP	FIELD	1	Routing Indicator Type Code
RI	GROUP	7	Routing Indicator
RI-SW	GROUP	4	
RI-C 1	FIELD	1	
RI-C2-C4	FIELD	3	
RI-TRIB	FIELD	3	
AREA-X	FIELD	1	Area Code
CMD	FIELD	4	Operating Command Code
ACTIV	FIELD	14	Activity
CLOC	FIELD	8	Location
FILLER	FIELD	2	State or Country Code
CARD-NR	FIELD	1	Card Number
CCSD	FIELD	8	Communication Circuit System Designator
EQUIP-CD	FIELD	6	Equipment Code
SPEED	FIELD	2	Circuit Speed Code
FORMT	FIELD	3	Message Format
SEC	FIELD	1	Security
OP-HR	FIELD	1	Hours of Operation Code
CHANR-A	FIELD	1	Channel Type
CHANR	FIELD	3	Channel Number
DESTNR	FIELD	3	Destination Number
SEC-RI-TYP	FIELD	1	Secondary RI Type Code
SEC-RI	GROUP	7	Secondary Routing Indicator
SEC-RI-SW	GROUP	4	
SEC-RI-C1	FIELD	1	

Table 10-3. AMIE Extract Record Format (Cont'd)

MNEMONIC REFERENCE	LEVEL	NUMBER OF CHARACTERS	ITEM DESCRIPTION
SEC-RI-C2-C4	FIELD	3	
SEC-RI-TRIB	FIELD	3	
LTC	FIELD	1	Line Traffic Controller
PDC	FIELD	4	
DATE-CD	FIELD	1	Date Code
FILLER	FIELD	42	

Table 10-4. AMIE Link File Format

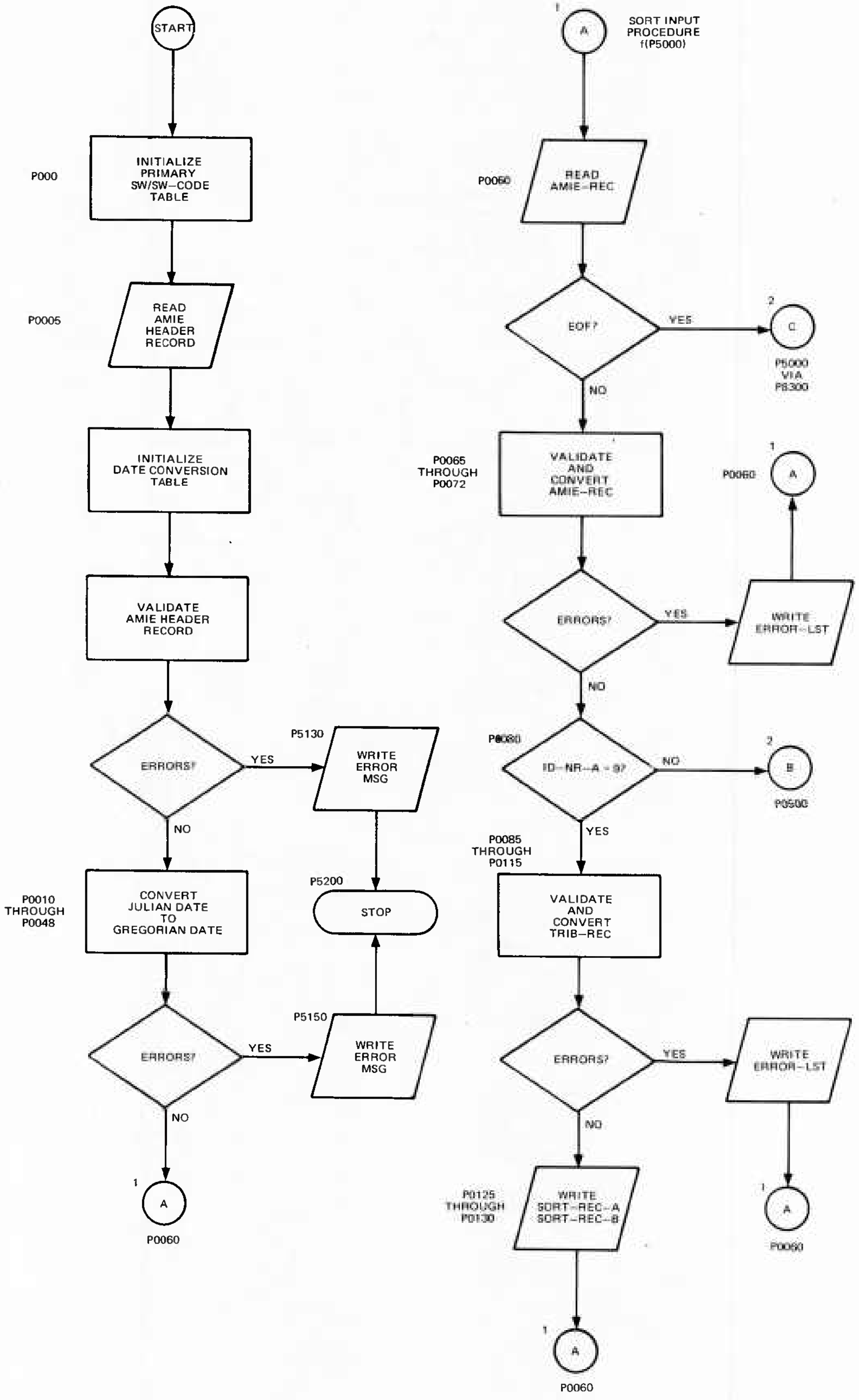
MNEMONIC REFERENCE	LEVEL	NUMBER OF CHARACTERS	ITEM DESCRIPTION
AMIELNK	FILE	80	AMIE Link File
LNKREC	RECORD	80	
SFLD-A	FIELD	4	Source Switch Designator
SFLD-C	FIELD	7	Destination Switch Designator
SFLD-E	FIELD	3	Channel Number
SFLD-F	FIELD	1	Line Traffic Controller
SFLD-H	FIELD	2	Link Speed Code
SFLD-I	FIELD	1	Link Security Code
FILLER	FIELD	62	Spaces

Table 10-5. AMIE Tributary File Format

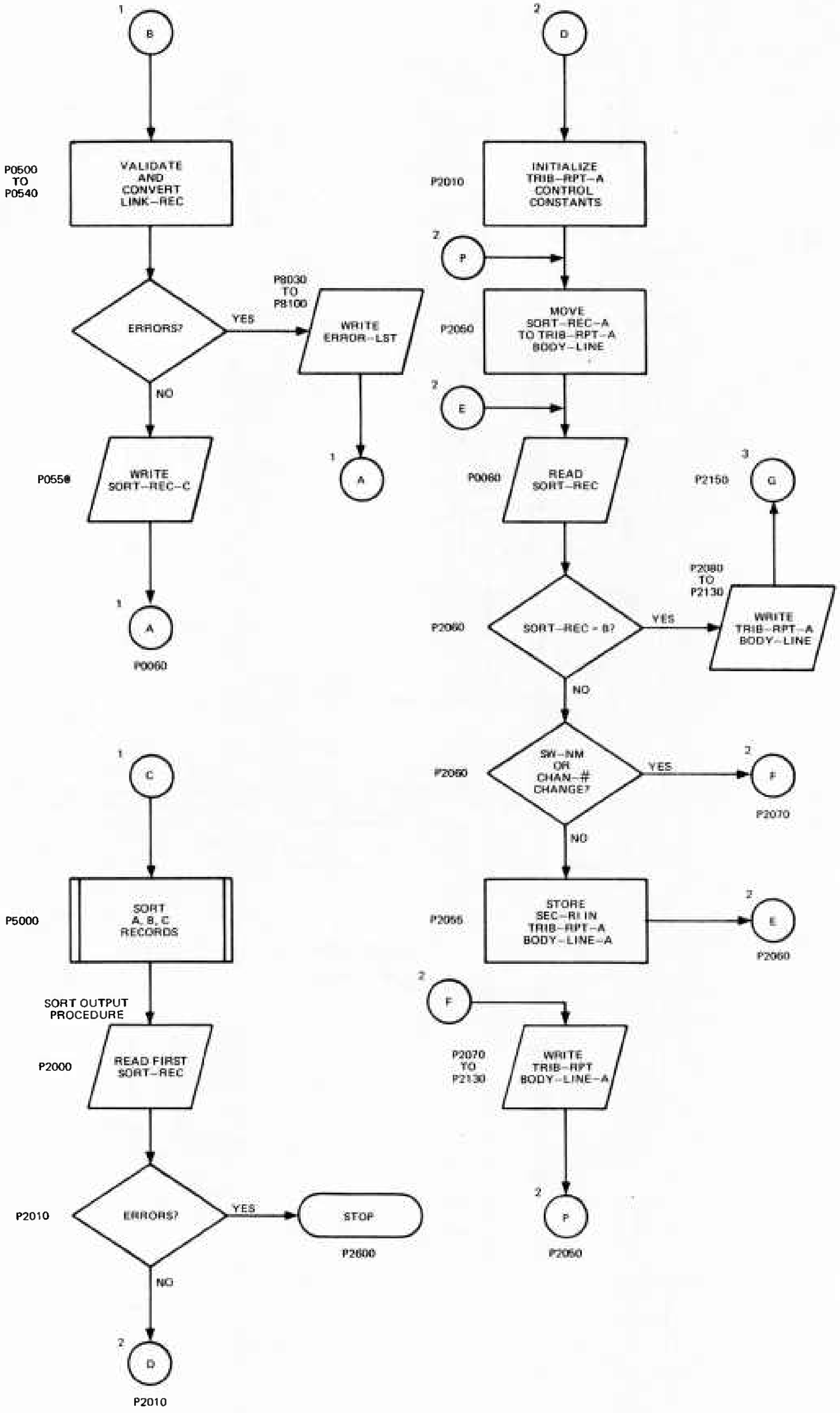
MNEMONIC REFERENCE	LEVEL	NUMBER OF CHARACTERS	ITEM DESCRIPTION
AMIETRIB	FILE	80	AMIE TRIB File
TRIBREC	RECORD	80	
SFLD-C	FIELD	7	Secondary Routing Indicator
SFLD-D	FIELD	4	Switch Designator
SFLD-E	FIELD	3	Channel Number
SFLD-F	FIELD	1	Line Traffic Controller
SFLD-G	FIELD	6	Equipment Code
SFLD-H	FIELD	2	Trib Speed Code
SFLD-I	FIELD	1	Trib Security Code
FILLER	FIELD	56	Spaces

Table 10-6. Sort File Record

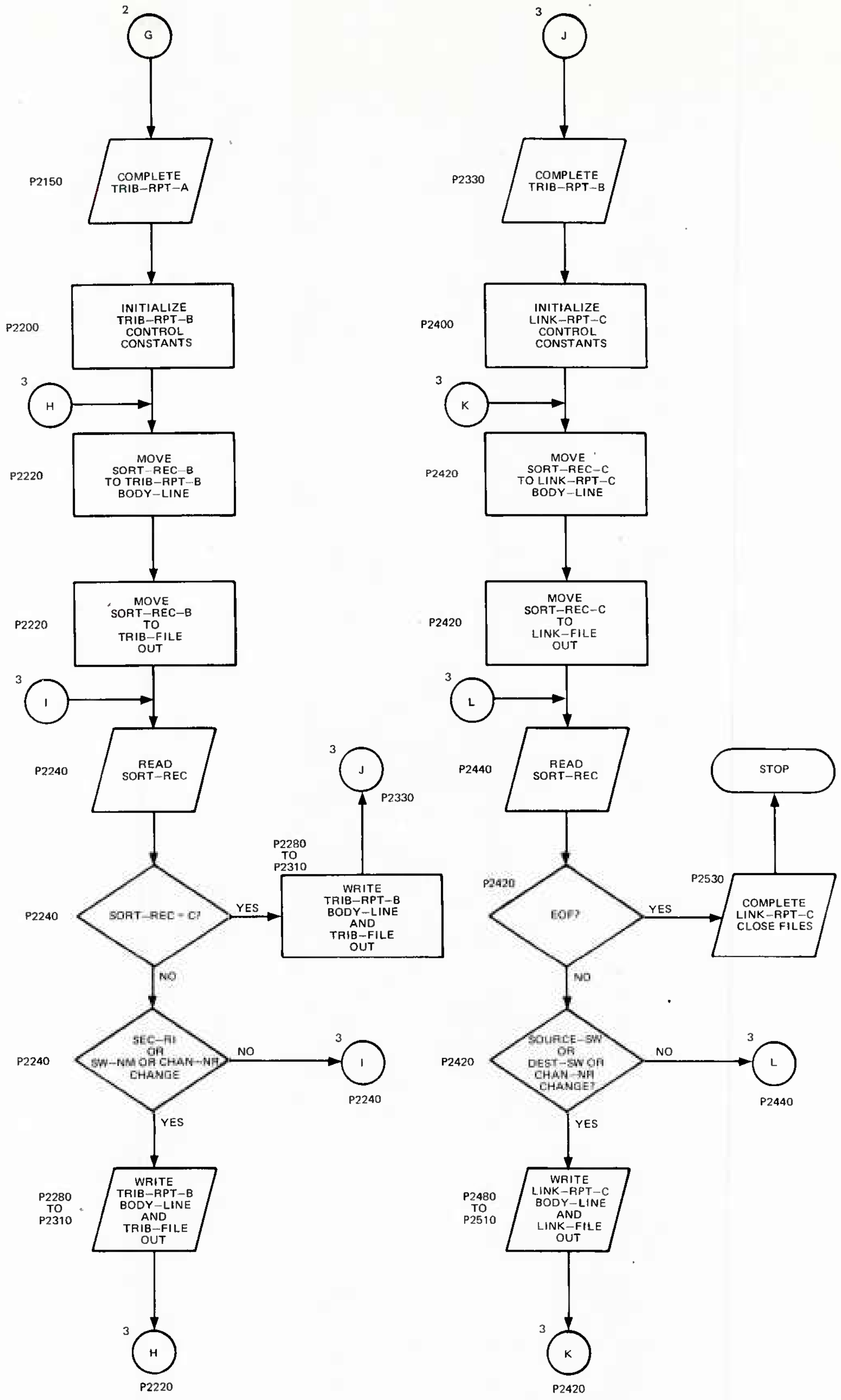
COLUMN	SORT RECORD FIELD	SORT SEQUENCE	A RECORD	B RECORD	C RECORD
1	PRI-KEY	1	A	B	C
2-5	SFLD-A	2	Switch		Origin Switch
6-8	SFLD-B	3	Channel		
9-15	SFLD-C	4	Secondary RI	Secondary RI	Destination Switch
16-19	SFLD-D	5		Switch	
20-22	SFLD-E	6		Channel	Channel
23	SFLD-F			LTC	LTC
24-29	SFLD-G			Equipment Code	
30-31	SFLD-H			Speed Code	Speed Code
32	SFLD-I			Security Code	Security Code



AMIE Extract Program (Sheet 1 of 3)



AMIE Extract Program (Sheet 2 of 3)


2
G
P2150
COMPLETE TRIB–RPT–A
P2200
INITIALIZE TRIB–RPT–B CONTROL CONSTANTS
3
H
P2220
MOVE SORT–REC–B TO TRIB–RPT–B BODY–LINE
P2220
MOVE SORT–REC–B TO TRIB–FILE OUT
3
I
P2240
READ SORT–REC
P2240
SORT–REC = C?
YES
P2280 TO P2310
WRITE TRIB–RPT–B BODY–LINE AND TRIB–FILE OUT
3
J
P2330
NO
P2240
SEC–RI OR SW–NM OR CHAN–NR CHANGE
NO
3
I
P2240
YES
P2280 TO P2310
WRITE TRIB–RPT–B BODY–LINE AND TRIB–FILE OUT
3
H
P2220
3
J
P2330
COMPLETE TRIB–RPT–B
P2400
INITIALIZE LINK–RPT–C CONTROL CONSTANTS
3
K
P2420
MOVE SORT–REC–C TO LINK–RPT–C BODY–LINE
P2420
MOVE SORT–REC–C TO LINK–FILE OUT
3
L
P2440
READ SORT–REC
P2420
EOF?
YES
P2530
COMPLETE LINK–RPT–C CLOSE FILES
STOP
NO
P2420
SOURCE–SW OR DEST–SW OR CHAN–NR CHANGE?
NO
3
L
P2440
YES
P2480 TO P2510
WRITE LINK–RPT–C BODY–LINE AND LINK–FILE OUT
3
K
P2420

AMIE Extract Program (Sheet 3 of 3)

SECTION 11 - ASSEMBLE TRAFFIC PROGRAM

11.1 PROGRAM REQUIREMENTS DESCRIPTION

11.1.1 Program Name: Assemble Traffic Program

11.1.2 Program Identification: DNASMTR1

11.1.3 Purpose

The Assemble Traffic program creates a Traffic file from the data contained in the Header Extract Tributary file. Each record written on the Traffic file contains the origin, destination, and other characteristics of one message segment. In addition, the program provides a printed summary of the traffic sorted by switch and channel number (Assemble Traffic Program Summary Report), and a file of tributary channels for input to the Simulator's Status Generator program. Each record on the Tributary file is listed with a sequence number on the Tributary File Listing.

11.1.4 Requirements

The Assemble Traffic program is written in ANSI COBOL and requires 17K bytes of core when executed on an IBM 360/65.

11.2 INPUT SPECIFICATIONS

Input to the Assemble Traffic program consists a set of switch-RADAY cards, the AMIE Tributary file, and the Header Extract Tributary file.

11.2.1 Switch-RADAY Card Set

The switch-RADAY card set communicates to the Assemble Traffic program the actual RADAY for which traffic is supplied for a given simulation. It is possible that the RADAY for data submitted by one or more switches could be for other then the standard RADAY. The program uses the card set to ensure that only traffic for a specified day for each switch is used.

The standard RADAY for the simulation is identified on the first card of the switch-RADAY card set. The format for this card is illustrated in Table 11-1. One card for each switch in the network follows the standard RADAY card. These cards identify the sample day for the indicated switch. The format for these cards is also illustrated in Table 11-1. The card sequence following the standard RADAY card may be in any order.

11.2.2 AMIE Tributary File

The AMIE Tributary file, produced as output by the AMIE Extract program, is read into an internal table by the Assemble Traffic program. This AMIE tributary data is used to resolve unmatched Header Extract tributary-to-link and link-to-tributary records. Records are sorted on this file in secondary routing indicator, primary switch, and channel number sequence.

The file is disk resident and sequentially organized with 80-character (EBCDIC) logical records and 100 logical records per block. The record format for this file is illustrated in Table 10-5.

11.2.3 Header Extract Tributary File

The Header Extract Tributary file is produced as an output by the Header Extract program. Each record on this file describes the characteristics of a message segment which originated or terminated at a tributary, or passed directly from one tributary to another within a switch. The program examines each record in terms of RADAY, and, if found acceptable, uses the record to create the Traffic and Tributary files.

The Header Extract Tributary file is organized sequentially on tape with 61-character (EBCDIC) logical records and 70 logical records per block. When the program is run with a full RADAY sample from all primary switches, the Tributary file contains in excess of 500,000 records on two reels of 1600-BPI magnetic tape. The record format for the file is illustrated in Table 9-4.

11.3 OUTPUT SPECIFICATIONS

Output from the Assemble Traffic program is composed of the Traffic and Tributary files, and three listings: the Tributary File Listing, the Assemble Traffic Program Summary Report, and an Error Listing containing certain rejected Header Extract Tributary file input records.

11.3.1 Traffic File

Each record in the Traffic file contains a description of one message segment which passed through the AUTODIN system. The origin and destination is specified, as are the characteristics (line block count, security, precedence, language media format, and station serial number). The time of transmission (TOT), or that time at which the message entered the AUTODIN system, is also included in the record. The Speed of Service (SOS) (difference between the end of message out time at the destination switch and the TOT time) is computed in seconds and inserted in each output record. A"-1" is entered into the SOS field for those records where the computation is not possible.

The Traffic file is organized sequentially on disk with 33-character (EBCDIC) logical records and 1000 records per block. The record format is illustrated in Table 11-2.

11.3.2 Network Tributary File

The Network Tributary file contains one record defining each unique tributary within the AUTODIN system as identified by an analysis of the origins and destinations of messages. Each output record specifies the switch to which the tributary is tied, the maximum security capability, and the Line Traffic Controller (LTC) number. An average queue delay in seconds is computed for each tributary. This value is based on low precedence messages which are delivered to the tributary. The record layout is described in Section 2.

Each record is capable of containing four types of message delay values:

1. Teletype Transmission Delay
2. Punched Card Transmission Delay
3. Magnetic Tape Transmission Delay
4. End of Message Acknowledgement Delay

Currently, only the appropriate Teletype Transmission Delay is inserted into each output record by the program.

The network Tributary file is organized sequentially on disk with 80-character (EBCDIC) logical records and 10 records per block. The record format is illustrated in Table 11-3.

11.3.3 Tributary File Listing

Each line on this listing represents one record on the Network Tributary file. A sequence number identifies the position of the record in the file.

11.3.4 Assemble Traffic Program Summary Report

This report summarizes the number of messages which originated and terminated at each tributary during the sample RADAY period. The primary switch, channel number, routing indicator, and speed of each tributary is specified. The report is sorted by primary switch and channel number.

11.3.5 Error Listing

Certain types of data conditions or data errors which occur during the processing of the Header Extract Tributary file result in the tributary record being printed in the Error Listing together with a message. A list of error messages which might appear on this report are listed in Paragraph 13.5.2 of the User's Manual.

11.4 PROGRAM LOGIC

11.4.1 Introduction

Each record on the Header Extract Tributary file contains a condensed summary of a message transaction which occurred at an AUTODIN switch. Basically, the record tells how and from whom the message entered the switch and how and to whom the message left the switch. A message generally enters a switch from one of two sources; a message originated by a tributary entering AUTODIN for the first time, or a message being forwarded from another AUTODIN switch on an interswitch link. Similarly, messages leaving a switch are either being delivered to a tributary or being forwarded to another AUTODIN switch.

The function of the Assemble Traffic program is to determine the traffic existing in AUTODIN on the day the Header Extract sample was taken. It accomplished this by using the records which show originated messages and those which show delivered messages; sorting and matching them into a single record with the origin, destination, and other vital statistics. Identification of these records is done by using the Input and Output Channel Types. Any designation other than H or Z in these fields indicates a tributary connection.

The records of interest in the Header Extract Tributary file fall into three categories:

1. Tributary-to-tributary records indicated by other than H or Z in both the Input and Output Channel Types. This record shows a complete delivery and requires no further processing.

2. Tributary-to-link records indicated by other than H or Z in the Input Channel Type and by H or Z in the Output Channel Type. This record indicates an originated message but not the final destination.

3. Link-to-tributary records indicated by H or Z in the Input Channel Type and by other than H or Z in the Output Channel Type. This record indicates a delivered message but not the origin.

The program reads the Header Extract Tributary file input records which have been verified and filtered by the Header Extract program and separates the records into the preceding categories. The tributary-to-tributary records are complete and are written directly on the output Traffic file. The tributary-to-link and link-to-tributary records are sorted by their identifying characteristics and matched, where possible, to complete a record. When a match cannot be made, as when a complete sample is not available, an attempt is made to complete the record using the AMIE Tributary file.

11.4.2 Program Execution

As previously explained, the primary function of the Assemble Traffic program is to generate a Traffic file containing one complete record for each unique message segment identified in the Header Extract Tributary file. In addition, the program reconstructs the network tributary configuration based on the origin and destination tributaries specified in each of the message header records. In conjunction with the latter function, the program produces a summary report describing tributary activity. The program logic which accomplished this consists of eight segments, two primary internal tables, and three intermediate disk work files. The tables and their functions are:

1. SW-RADAY-TAB - This table contains the switch-RADAY card set supplied as parametric input to the program. There is one record initialized in the table for each primary switch in the card set. Each record contains a one-character alphabetic code corresponding to the primary switch designator. In addition, each record contains a flag position indicating whether the RADAY for the respective switch is standard, nonstandard, or not specified in the switch-RADAY card set.

2. AMIE-TAB - This internal program table holds all records contained on the AMIE Tributary file. The table is in sort order by secondary routing indicator, switch designator, and channel number. It is used to complete origin and destination message records when their respective destination and origin component records are not contained on the Header Exgract Tributary file.

The following temporary, disk resident work files are used by the program:

1. ORIG FILE - This file is used to sort all incomplete origin records.
2. DEST FILE - This file is used to sort all incomplete destination records.
3. DOFILE - The program writes two records on this file for each complete record written on the Traffic file. One identifies the origin and the other the destination of a completed message. These records, after being sorted, are used to generate the Network Tributary file and the Assemble Traffic Program Summary Report.

The program logic executes sequentially in eight functional areas:

1. The switch-RADAY card set is read, validated, and loaded into the SW-RADAY-TAB. The sample day for each switch is compared to the standard RADAY (the Julian day value appearing on the first card of the switch-RADAY card set). If the sample day is equal to the standard RADAY, the RADAY-TYPE field for the switch is set to "S". If the RADAY values differ, the field is set to "N", for nonstandard RADAY.
2. Each record in the AMIE Tributary file is read, validated, and loaded into the AMIE-TAB. The data in this table is used by the program to resolve unmatched origin and destination records.

3. Due to the large size of the AMIE-TAB table (up to 3100 records, 80-characters each), a modified binary search was included to provide rapid access to each record and to minimize core requirements. A count is maintained of the number of records read into AMIE-TAB. This count, when divided by two and truncated, serves as the first binary search subscript value in the BS-SS-TAB. Succeeding values in this table represent the preceding value divided by two and truncated.

   The binary search logic (P15000-BIN-SER) attempts to match the routing indicator value provided by the main program logic with a secondary routing indicator value contained in AMIE-TAB. Match and no-match conditions are indicated via the AMIE-BS-SER-FLG being set to 1 and 0 respectively. When a match condition occurs, the value COMP-SS points to the AMIE-TAB record location.

   In certain cases, the program logic requires that a match be made between an AMIE-TAB record and a record being processed based on both routing indicator and switch. The program uses serial scan logic (P1600-SER-SCAN) to find the required dual match condition. The binary search logic is first used to obtain a location within the specified routing indicator block. This pointer is then used by the logic to calculate the upper and lower boundary of a block of records to be searched for the dual match condition.

4. The fourth major segment of the program reads the Header Extract Tributary file records and selects those records containing a RADAY equal to that contained in the internal SW-RADAY-TAB for the reporting switch. Each record is categorized into one of three groups and processed accordingly, as follows:

   a. <u>Tributary-to-Tributary or Storage (SVM)-to-Tributary</u> - Records in this category are complete and released to both the Traffic file and the DOFILE if they pass additional editing criteria.

b. Tributary-to-Storage or Storage (SVM)-to-Link - Records in this category are defined as incomplete origin records and are written on the ORIG FILE for matching with corresponding records contained on the DEST FILE in a later segment of the program; additional editing based on RADAY type is accomplished on records falling into this category prior to their release to the ORIG FILE: all nonstandard RADAY records destined for another switch (i.e. interswitch records) are eliminated; nonstandard RADAY intraswitch records are retained.

c. Storage-to-Tributary or Link-to-Tributary - Records in this category are defined as incomplete destination records and are written on the DEST FILE for matching with corresponding records contained on the ORIG FILE in a later segment of the program; additional editing based on RADAY type is accomplished on records falling in this category prior to their release to the DEST FILE: all nonstandard RADAY destination records originating from any standard RADAY switch are eliminated; intraswitch destination records are retained; all nonstandard RADAY destination records originating from a nonstandard RADAY switch are completed using AMIE-TAB data and released to both the traffic file and the DOFILE.

The intraswitch test logic in a and b is based on two tests:

(1) The primary switch (C1-C4) of the OSRI field containing the same primary switch value (C1-C4) as the DSRI field.

(2) The DSRI in an origin record containing the same primary switch in AMIE-TAB as the record reporting-switch field or, the OSRI in a destination record containing the same primary switch in AMIE-TAB as the record reporting-switch field.

The OSRI or DSRI may appear several times in AMIE-TAB under different primary switch values (i. e. multiple-homed tributaries). If the OSRI or DSRI is a minor switch (C1-C4 not equal to a primary switch value) it is necessary to blank out the low order three characters of the field prior to searching AMIE-TAB for the respective value. This is because all minor switch routing indicators are carried in AMIE-TAB in this form.

When an origin or destination tributary cannot be found in AMIE-TAB, the record is printed in the Error List with an appropriate message.

5. The ORIG FILE and DEST FILE are sorted following completion of AMIE Tributary file processing. The sort sequence for these files is:

    Origin Routing Indicator
    Time of Transmission
    Serial Number
    Destination Routing Indicator

6. The sixth major segment of the program contains the ORIG FILE and DEST FILE match logic. When a match occurs, a message record is constructed and released to the Traffic file. In addition, origin and destination records are built and released to the DOFILE. The program attempts to complete unmatched records with information from AMIE-TAB. When an origin or destination tributary cannot be found in AMIE-TAB, the record is printed in the Error List with an appropriate message. Records which are completed via AMIE-TAB are used to build a Traffic file record and DOFILE records.

7. The DOFILE destination and origin records are sorted following completion of ORIG FILE and DEST FILE processing. The sort sequence for this file is:

    Primary switch
    Channel number
    Routing indicator

8. The final segment of the program produces the Assemble Traffic Summary Report, the Tributary file, and the Sequenced Tributary file listing. Each of these outputs is produced concurrently from the sorted DOFILE.

   The Assemble Traffic Summary Report is computed by retaining counts of the input sort sequence is in the required report order. The Tributary file is created in a like manner. The average starting queue for low priority messages in seconds is computed from the DOFILE destination records and included in each Tributary file record.

   The speed for each Tributary file record is not always obtained from AMIE-TAB. If the tributary is not located in the table via a binary search followed by a serial scan of the routing indicator block for desired channel, a second search is initiated. The second search is a serial scan of the entire AMIE-TAB based on the primary switch and channel number. If this scan does not yield the desired channel, the speed of the tributary is obtained from the internal program SPEED-TAB and is based on the tributary channel type code.

11.4.3 Internal Primary Switch Table Modification

The program contains an internal table (SW-RADAY-TAB) which lists all valid AUTODIN switches. The table is capable of containing 24 switch values, but is currently initialized with 21 switches. If the table is modified, the field SW-COUNT-PLUS1 must be set to the number of values in the table plus one. Switches may appear in any order.

11.4.4 Error Diagnostics and Messages

1. AMIE FILE LENGTH GT 3100 RECS (COUNT) - The AMIE file contains a record count greater than 3100 records. Since the program is designed to terminate execution if over 3100 records are present, the job must be rerun. Prior to resubmission of the job, decrease the AMIE tributary file size or increase the Assemble Traffic program AMIE Tributary Table size.

2. AMIE FILE CONTAINS 0 RECORDS - The program terminated execution as a result of finding zero records on the AMIE Tributary file. Resubmit the

job with an AMIE Tributary file that contains at least one record but less than 3101 records.

3. NO RECS IN DOFIEL ON OUTPUT - The program determined that there were zero valid tributary messages on the Header Extract Tributary file. Compare the switch-RADAY card set against the Header Extract Summary Report. If this does not explain why the Header Extract Tributary file records were rejected, examine the Error Listing. If neither of these steps provides a solution to the problem, obtain a dump of the first 1000 records of the Header Extract Tributary file.

Program logic flow charts are included at the end of this section.

11.4.5 Memory Layout

The program does not use subroutines and requires a single block of memory for executable instructions and storage arrays.

11.5 MACHINE DEPENDENCE

In general, the Link Load program is machine independent. All coding conforms to ANSI Standards. However, in implementing the program on an IBM 360/65, some machine dependence was unavoidable. The Identification Division describes an IBM 360/65.

Table 11-1. Standard RADAY and Switch-RADAY Card Formats

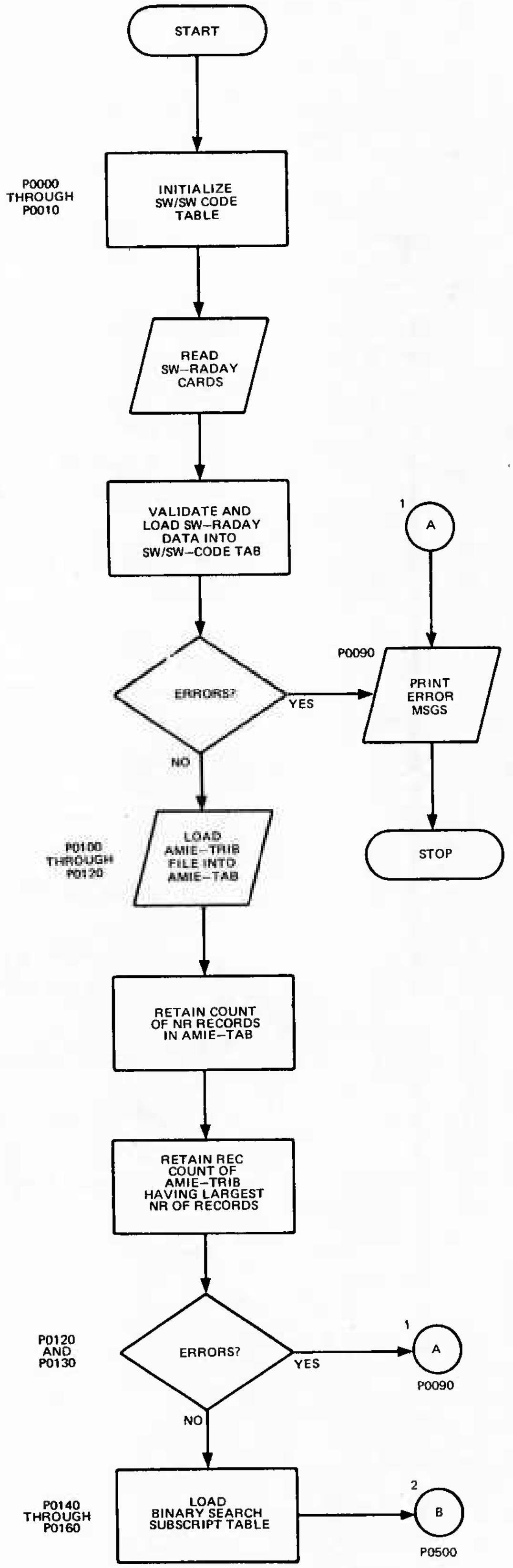
CARD COLUMN	DESCRIPTION	EXPLANATION
STANDARD RADAY CARD FORMAT		
1 - 5	Blank	
6 - 8	Standard RADAY	Numeric Julian Day between 001 and 366
9 - 80	Blank	
SWITCH-RADAY CARD FORMAT		
1 - 4	Primary Switch	
5	Blank	
6 - 8	RADAY for sample submitted for switch in cc 1 - 4	Numeric Julian Day between 001 and 366
9 - 80	Blank	

Table 11-2. Traffic File Record Format

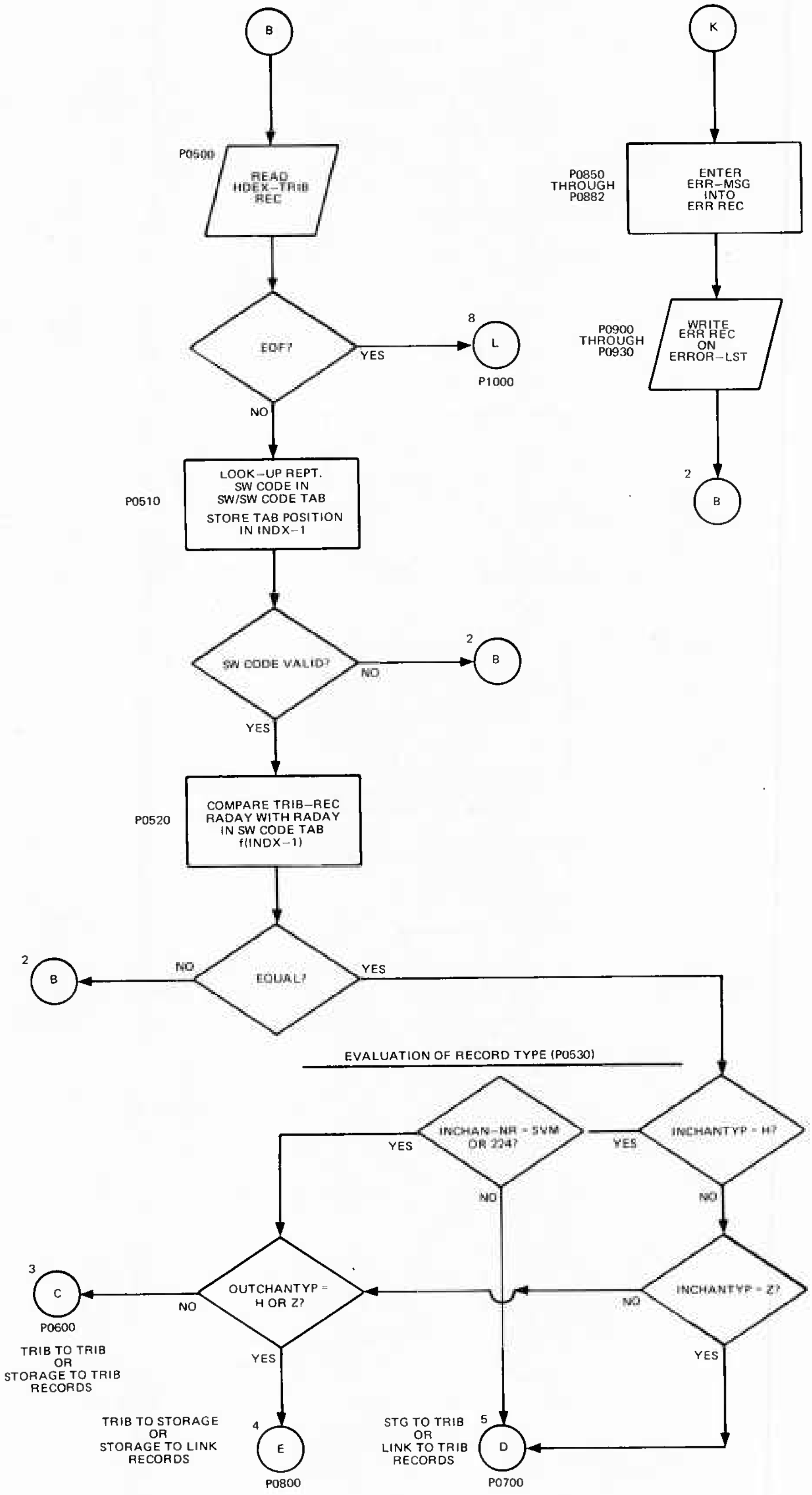
MNEMONIC REFERENCE	LEVEL	NUMBER OF CHARACTERS	ITEM DESCRIPTION
HEXMSG	FILE	33	Traffic File
HEXMSG-REC	RECORD	33	
RADAY	GROUP	3	Time of Transmission: Day
RADAY-C1, C2	FIELD	2	
RADAY-C3	FIELD	1	
PRECED	FIELD	1	Precedence
SECUR	FIELD	1	Security
LMF	FIELD	1	Language Media Format
ORIG-SW	FIELD	3	Message Origin Switch
INCHANR	FIELD	3	Input Channel to Origin Switch
DEST-SW	FIELD	3	Message Destination Switch
OCHANR	FIELD	3	Output Channel from Destination Switch
OSSN	FIELD	4	Origin Serial Number
LBC	FIELD	3	Line Block Count
TOT	GROUP	4	Time of Transmission
TOT-HR	FIELD	2	TOT: Hour
TOT-MIN	FIELD	2	TOT: Minute
SOS	FIELD	4	Speed of Service in Minutes

Table 11-3. Tributary File Record Format

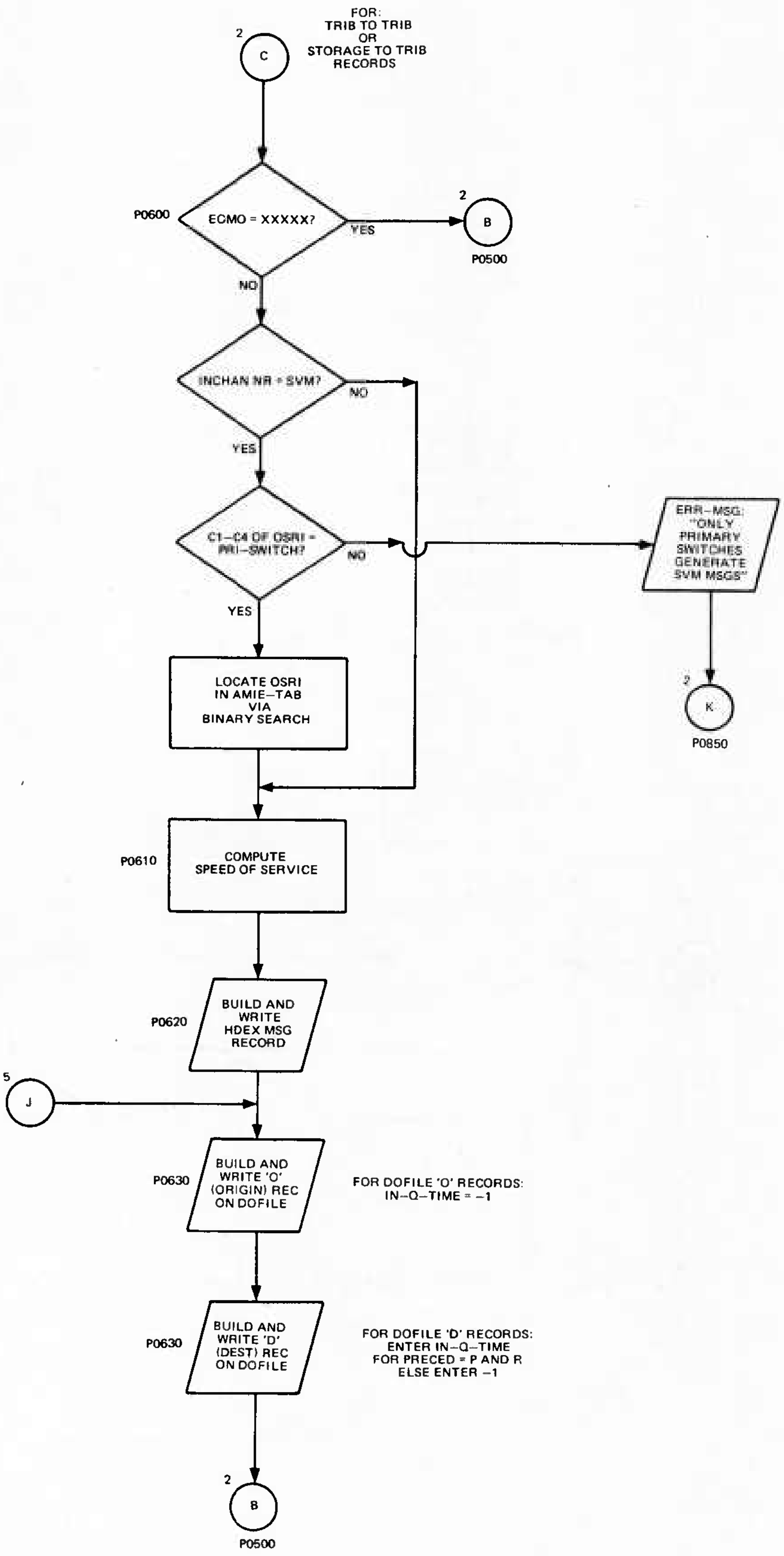
MNEMONIC REFERENCE	LEVEL	NUMBER OF CHARACTERS	ITEM DESCRIPTION
DINTRIB	FILE	80	Assemble Traffic TRIB File
DINTRIB-REC	RECORD	80	
FILLER	FIELD	9	Spaces
TRIB-ID	GROUP	6	Tributary Name
TRIB-SW	FIELD	3	Switch Servicing Tributary
TRIB-CHA	FIELD	3	Tributary Channel Number
FILLER	FIELD	1	Spaces
NR-TCHAN	FIELD	3	Number of Channels (001)
FILLER	FIELD	1	Spaces
MAX-SECUR	FIELD	1	Security Indicator (T)
FILLER	FIELD	3	Spaces
T-TX-D	FIELD	5	Teletype Transmission Delay
FILLER	FIELD	1	Spaces
PC-TX-D	FIELD	5	Punched Card Transmission Delay
FILLER	FIELD	1	Spaces
MT-TX-D	FIELD	5	Magnetic Tape Transmission Delay
FILLER	FIELD	1	Spaces
EOM-ACK-D	FIELD	5	End of Message Acknowledgement Delay
FILLER	FIELD	1	Spaces
NODE-NM-C1	FIELD	1	Switch Prefix (R)
NODE-NM	FIELD	3	Switch Servicing Tributary
FILLER	FIELD	3	Spaces
LTC	FIELD	1	LTC Number Serving Channel
FILLER	FIELD	1	Spaces
VON=DELAY	FIELD	5	AUTOVON Dial Delay (00000)
FILLER	FIELD	1	Spaces
AVG-Q-DELAY	FIELD	5	Average Starting Queue for Low Priority Messages in Seconds
FILLER	FIELD	12	Spaces



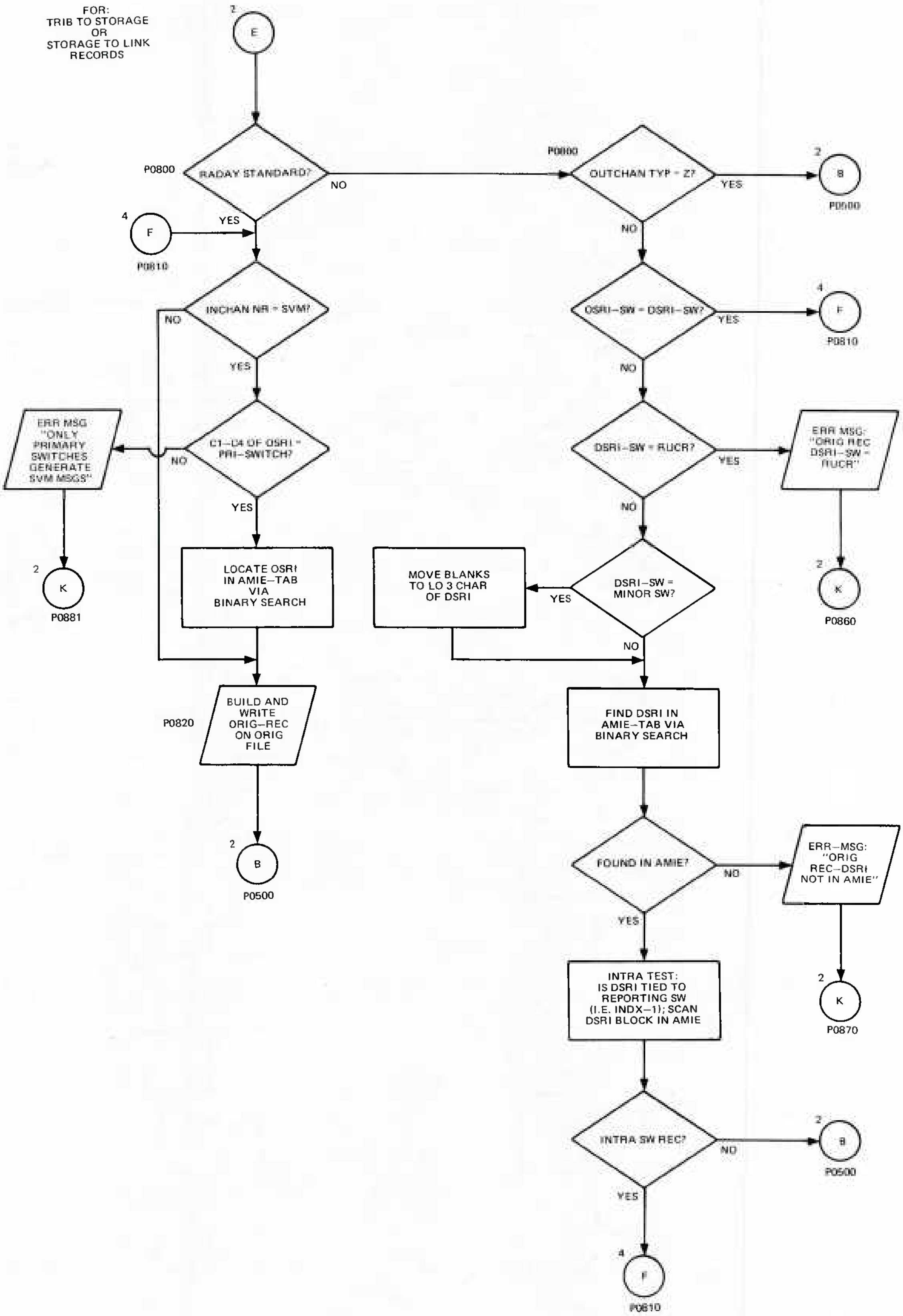
Assemble Traffic Program (Sheet 1 of 13)


B
P0500
READ HDEX–TRIB REC
EOF?
YES
8
L
P1000
NO
P0510
LOOK–UP REPT. SW CODE IN SW/SW CODE TAB
STORE TAB POSITION IN INDX–1
SW CODE VALID?
NO
2
B
YES
P0520
COMPARE TRIB–REC RADAY WITH RADAY IN SW CODE TAB f(INDX–1)
EQUAL?
NO
2
B
YES
EVALUATION OF RECORD TYPE (P0530)
INCHANTYP = H?
YES
INCHAN–NR = SVM OR 224?
YES
NO
NO
INCHANTYP = Z?
NO
YES
OUTCHANTYP = H OR Z?
NO
3
C
P0600
TRIB TO TRIB OR STORAGE TO TRIB RECORDS
YES
TRIB TO STORAGE OR STORAGE TO LINK RECORDS
4
E
P0800
STG TO TRIB OR LINK TO TRIB RECORDS
5
D
P0700
K
P0850 THROUGH P0882
ENTER ERR–MSG INTO ERR REC
P0900 THROUGH P0930
WRITE ERR REC ON ERROR–LST
2
B

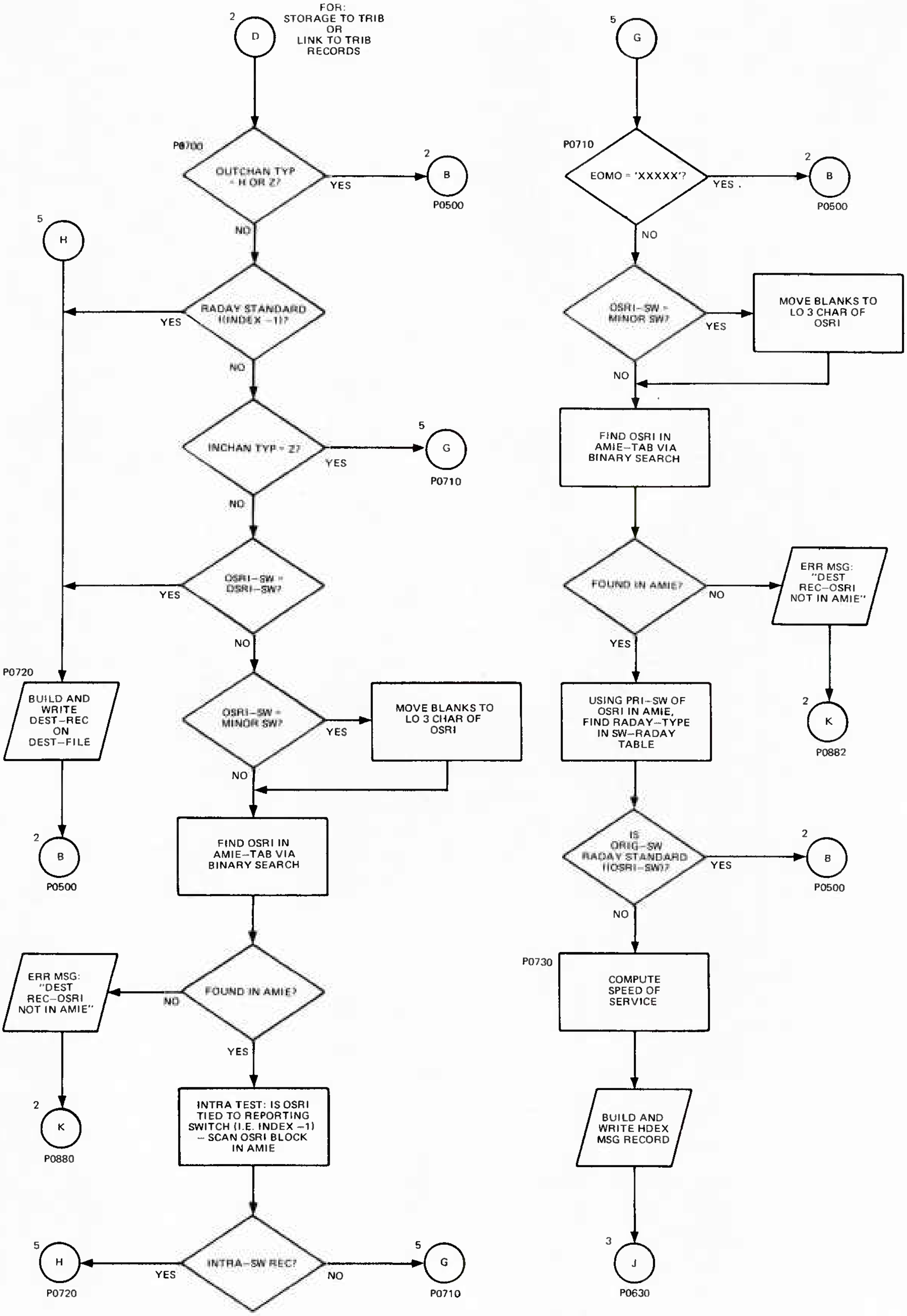
Assemble Traffic Program (Sheet 2 of 13)



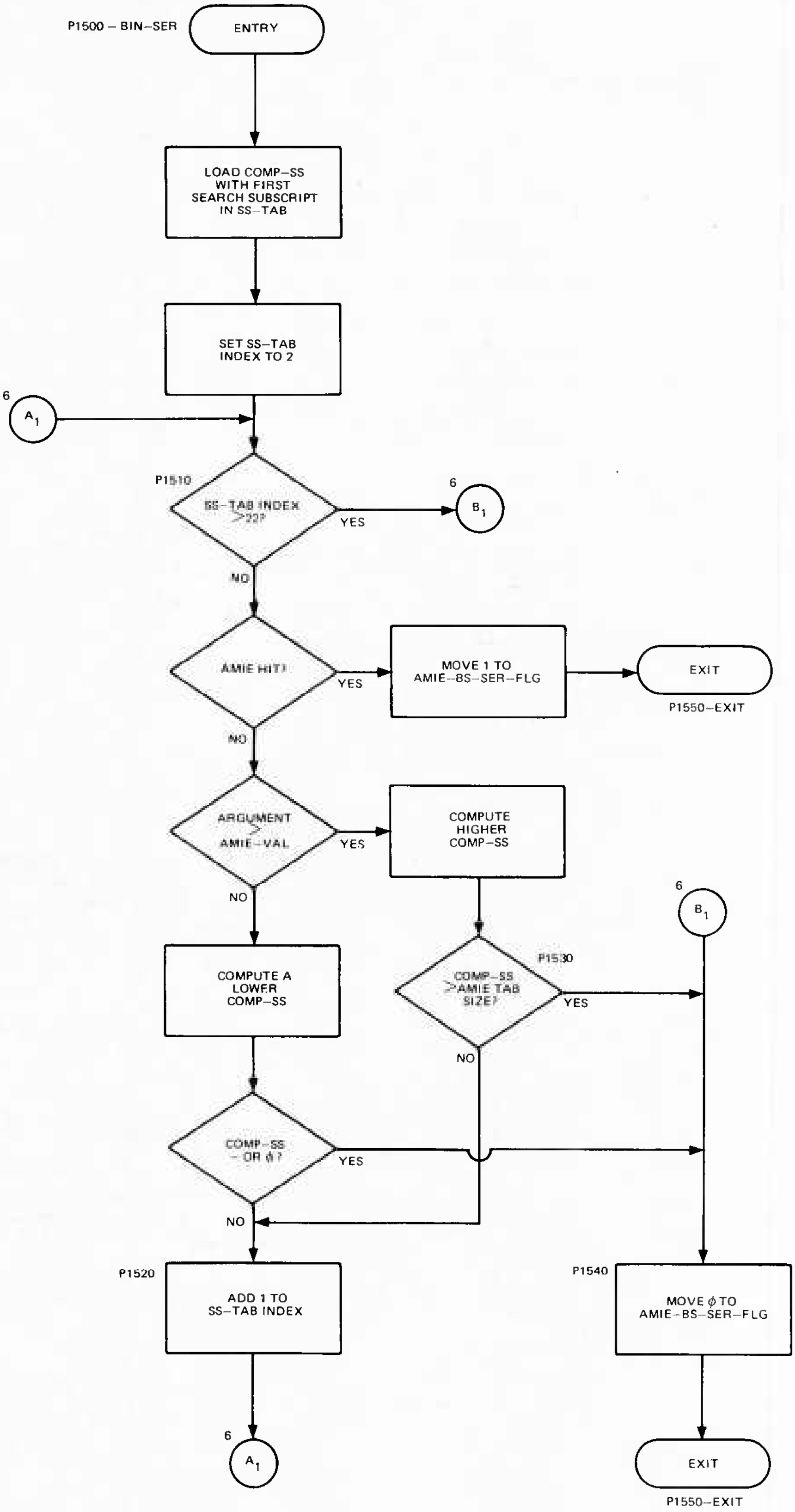
Assemble Traffic Program (Sheet 3 of 13)


FOR:
TRIB TO STORAGE
OR
STORAGE TO LINK
RECORDS
2
E
P0800
RADAY STANDARD?
NO
YES
4
F
P0810
INCHAN NR = SVM?
NO
YES
C1–C4 OF OSRI = PRI–SWITCH?
NO
YES
ERR MSG "ONLY PRIMARY SWITCHES GENERATE SVM MSGS"
2
K
P0881
LOCATE OSRI IN AMIE–TAB VIA BINARY SEARCH
P0820
BUILD AND WRITE ORIG–REC ON ORIG FILE
2
B
P0500
P0800
OUTCHAN TYP = Z?
YES
2
B
P0500
NO
OSRI–SW = DSRI–SW?
YES
4
F
P0810
NO
DSRI–SW = RUCR?
YES
ERR MSG: "ORIG REC DSRI–SW = RUCR"
2
K
P0860
NO
DSRI–SW = MINOR SW?
YES
MOVE BLANKS TO LO 3 CHAR OF DSRI
NO
FIND DSRI IN AMIE–TAB VIA BINARY SEARCH
FOUND IN AMIE?
NO
ERR–MSG: "ORIG REC–DSRI NOT IN AMIE"
2
K
P0870
YES
INTRA TEST: IS DSRI TIED TO REPORTING SW (I.E. INDX–1); SCAN DSRI BLOCK IN AMIE
INTRA SW REC?
NO
2
B
P0500
YES
4
F
P0810

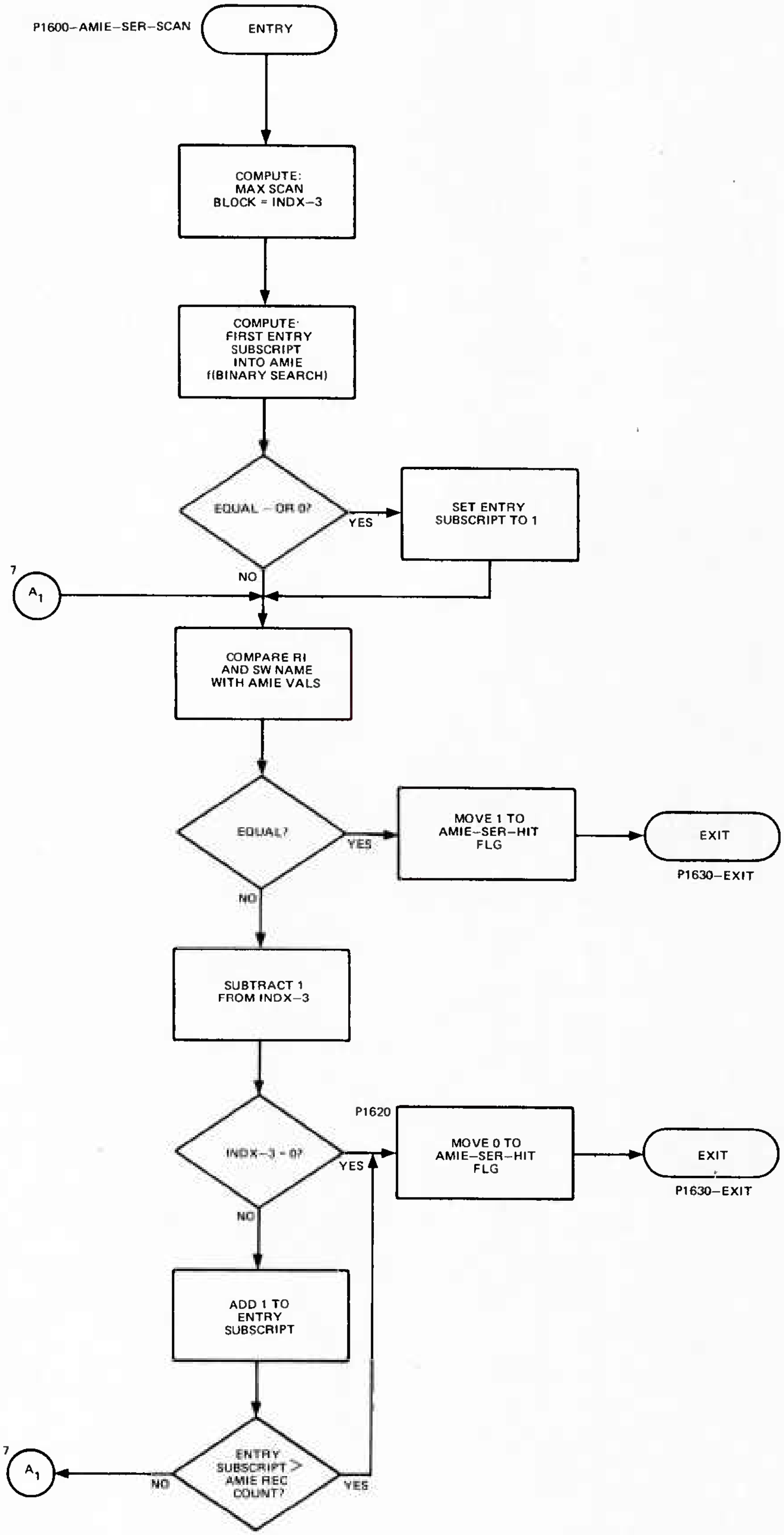
Assemble Traffic Program (Sheet 4 of 13)


2
D
FOR:
STORAGE TO TRIB
OR
LINK TO TRIB
RECORDS
P0700
OUTCHAN TYP = H OR Z?
YES
2
B
P0500
NO
5
H
RADAY STANDARD (INDEX −1)?
YES
NO
INCHAN TYP = Z?
YES
5
G
P0710
NO
OSRI−SW = DSRI−SW?
YES
NO
P0720
BUILD AND WRITE DEST−REC ON DEST−FILE
2
B
P0500
OSRI−SW = MINOR SW?
YES
MOVE BLANKS TO LO 3 CHAR OF OSRI
NO
FIND OSRI IN AMIE−TAB VIA BINARY SEARCH
FOUND IN AMIE?
NO
ERR MSG: "DEST REC−OSRI NOT IN AMIE"
2
K
P0880
YES
INTRA TEST: IS OSRI TIED TO REPORTING SWITCH (I.E. INDEX −1) − SCAN OSRI BLOCK IN AMIE
INTRA−SW REC?
YES
5
H
P0720
NO
5
G
P0710
5
G
P0710
EOMO = 'XXXXX'?
YES .
2
B
P0500
NO
OSRI−SW = MINOR SW?
YES
MOVE BLANKS TO LO 3 CHAR OF OSRI
NO
FIND OSRI IN AMIE−TAB VIA BINARY SEARCH
FOUND IN AMIE?
NO
ERR MSG: "DEST REC−OSRI NOT IN AMIE"
2
K
P0882
YES
USING PRI−SW OF OSRI IN AMIE, FIND RADAY−TYPE IN SW−RADAY TABLE
IS ORIG−SW RADAY STANDARD (OSRI−SW)?
YES
2
B
P0500
NO
P0730
COMPUTE SPEED OF SERVICE
BUILD AND WRITE HDEX MSG RECORD
3
J
P0630

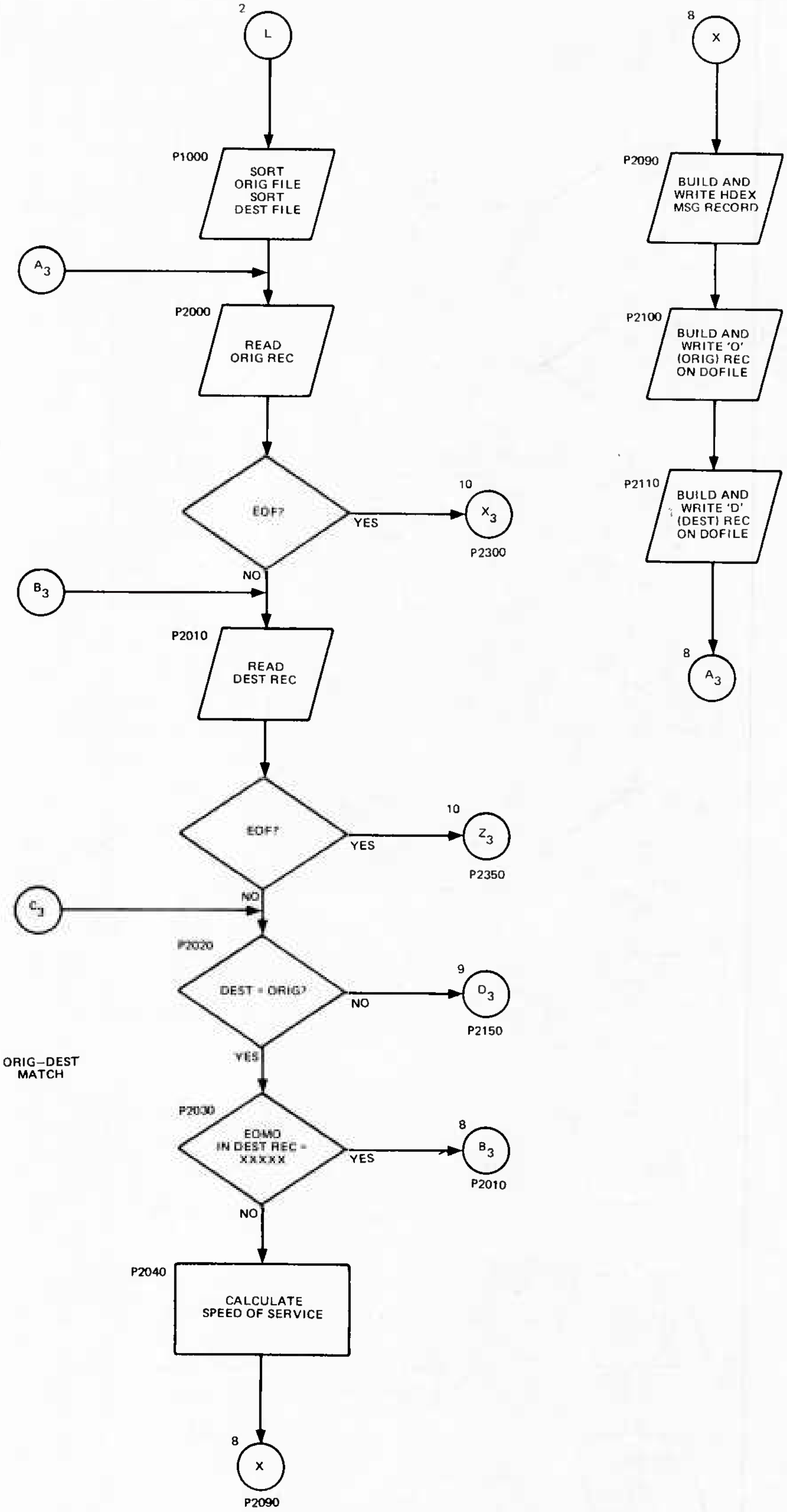
Assemble Traffic Program (Sheet 5 of 13)



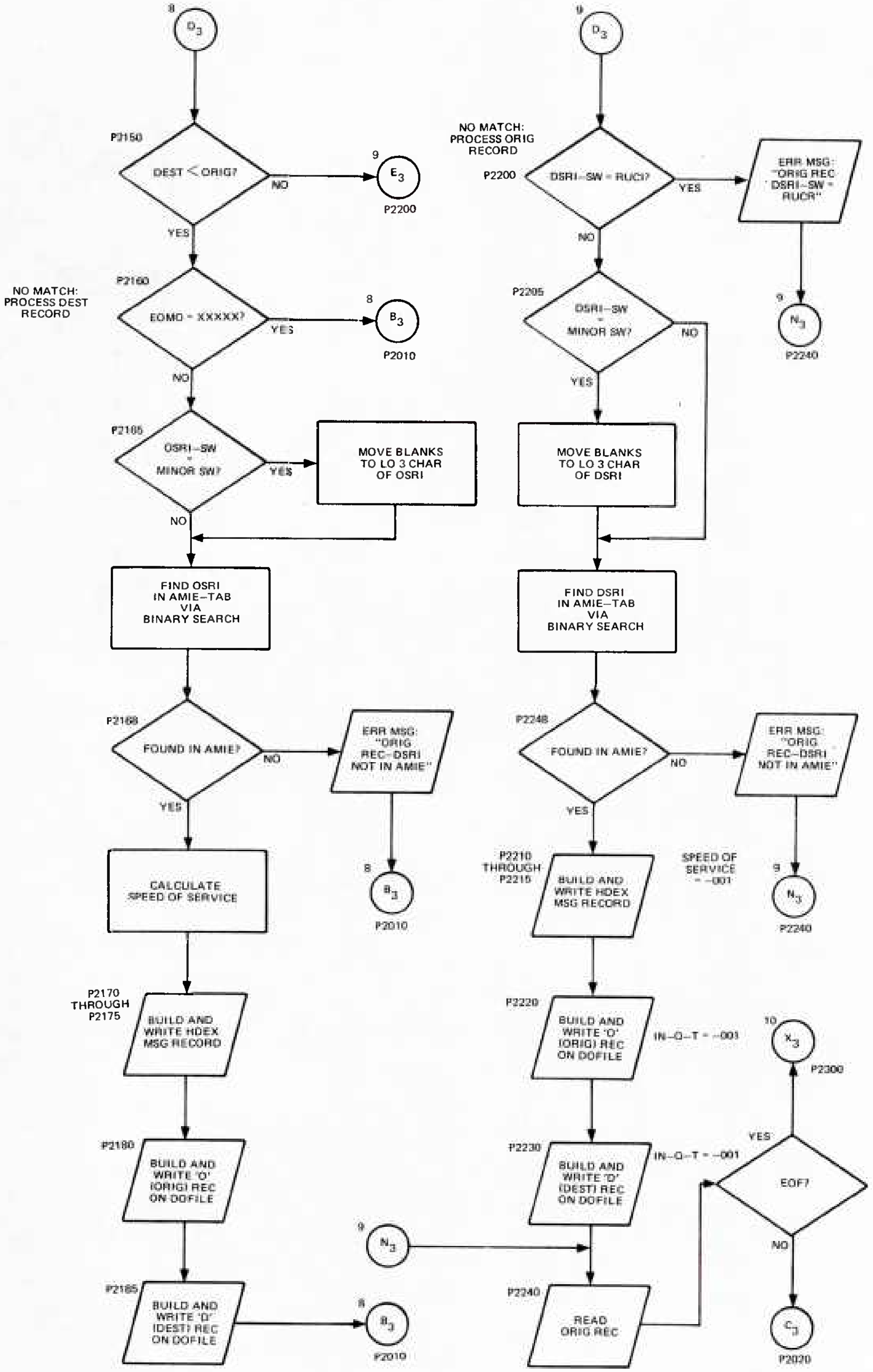
Assemble Traffic Program (Sheet 6 of 13)



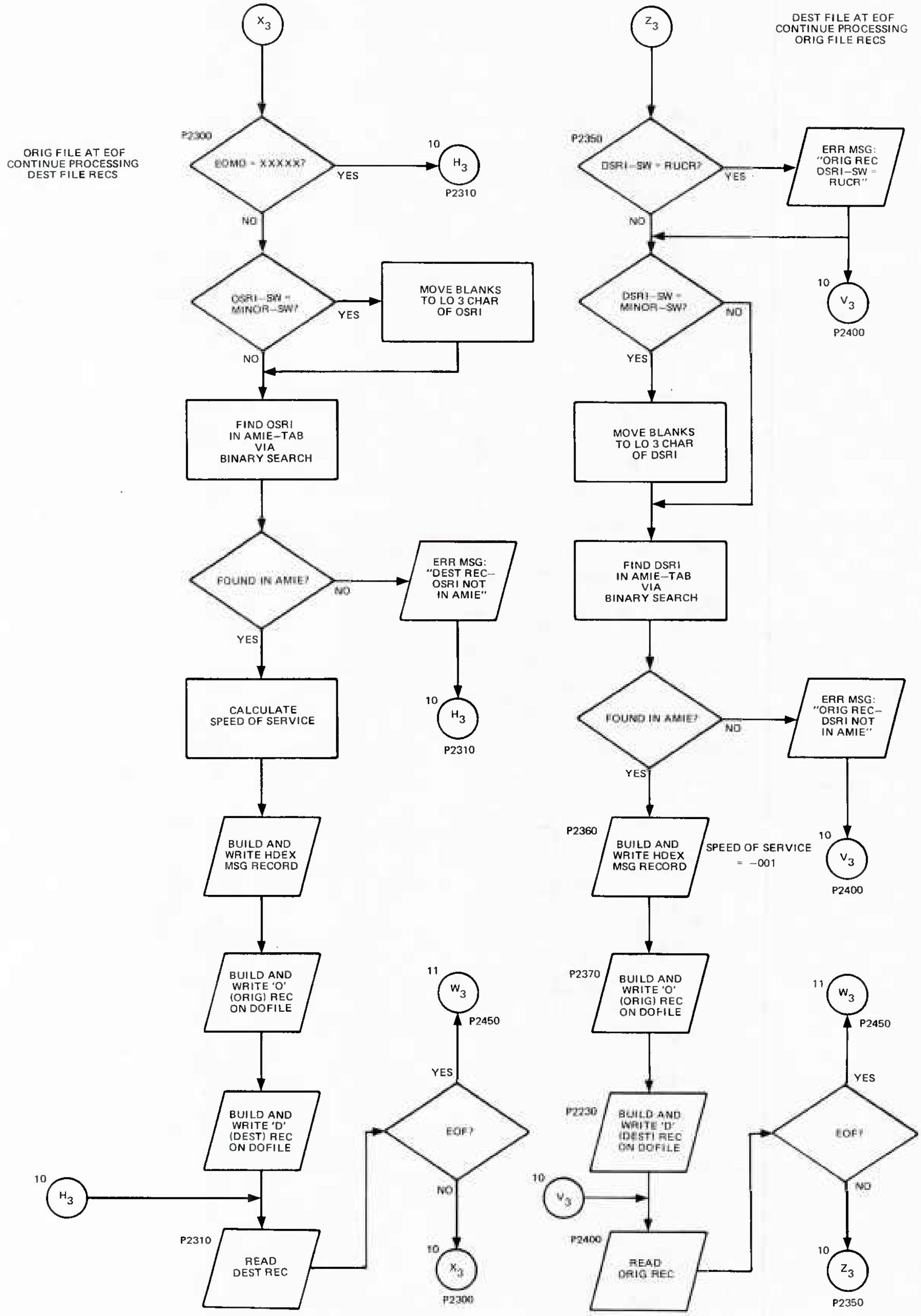
Assemble Traffic Program (Sheet 7 of 13)



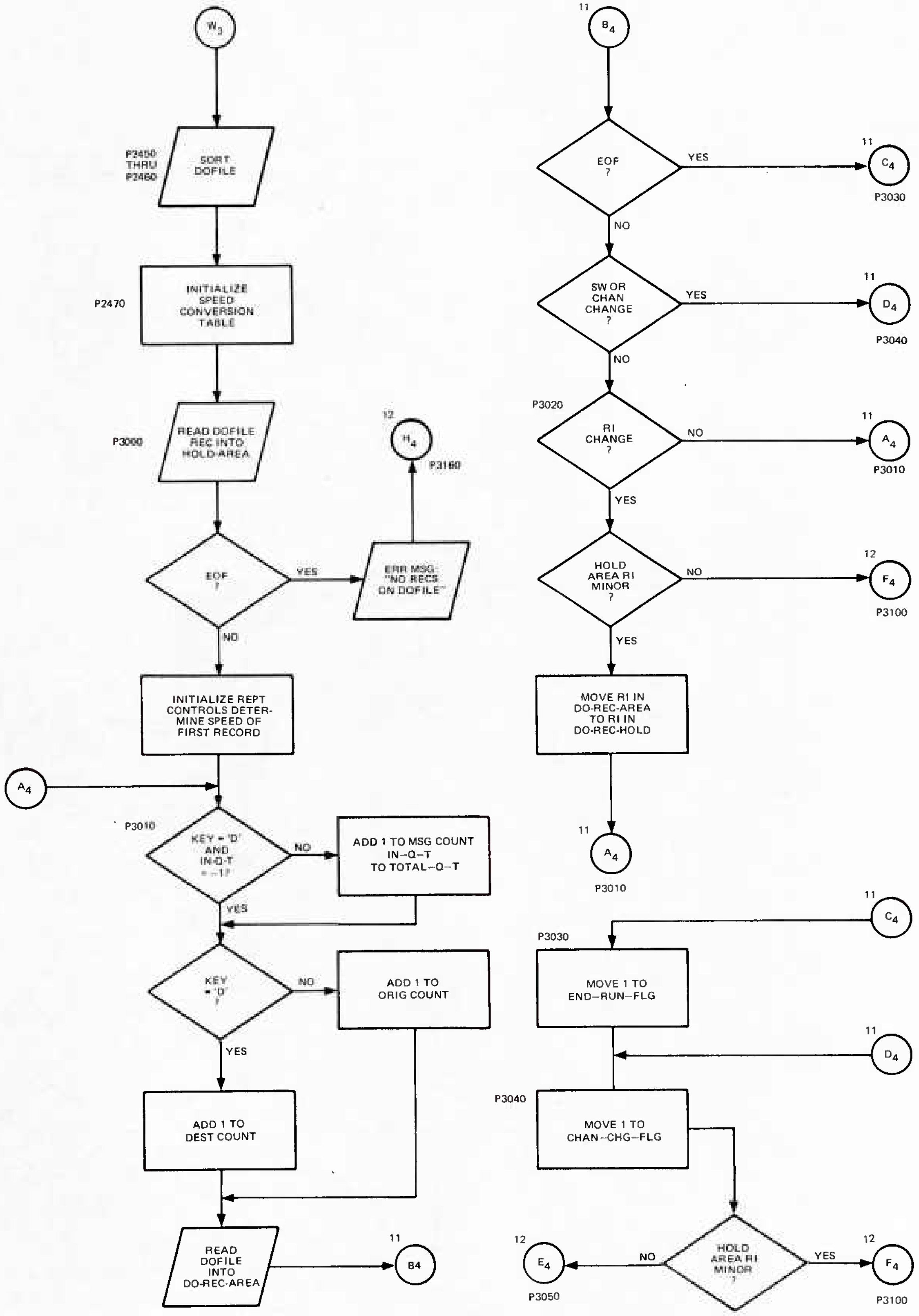
Assemble Traffic Program (Sheet 8 of 13)



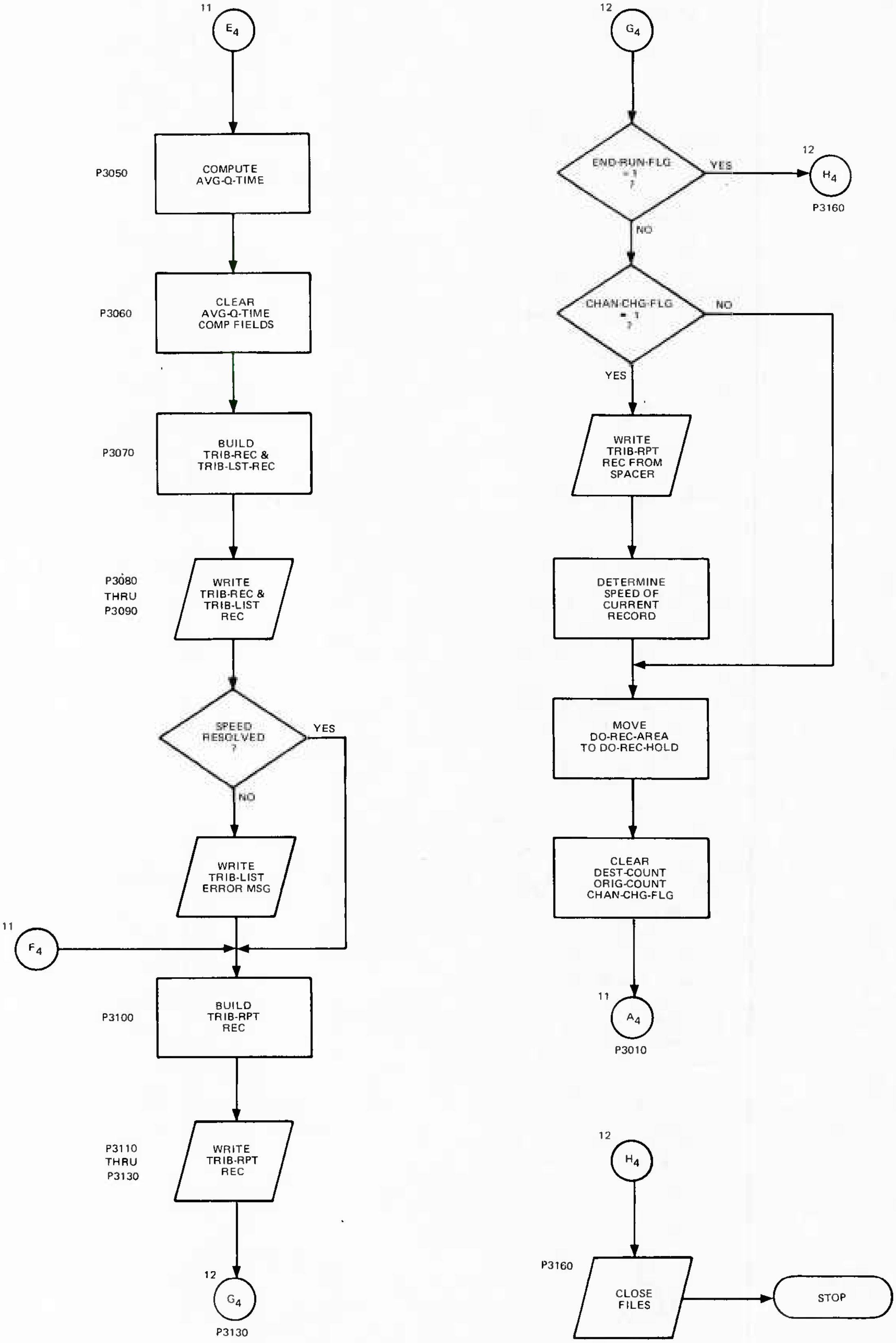
Assemble Traffic Program (Sheet 9 of 13)



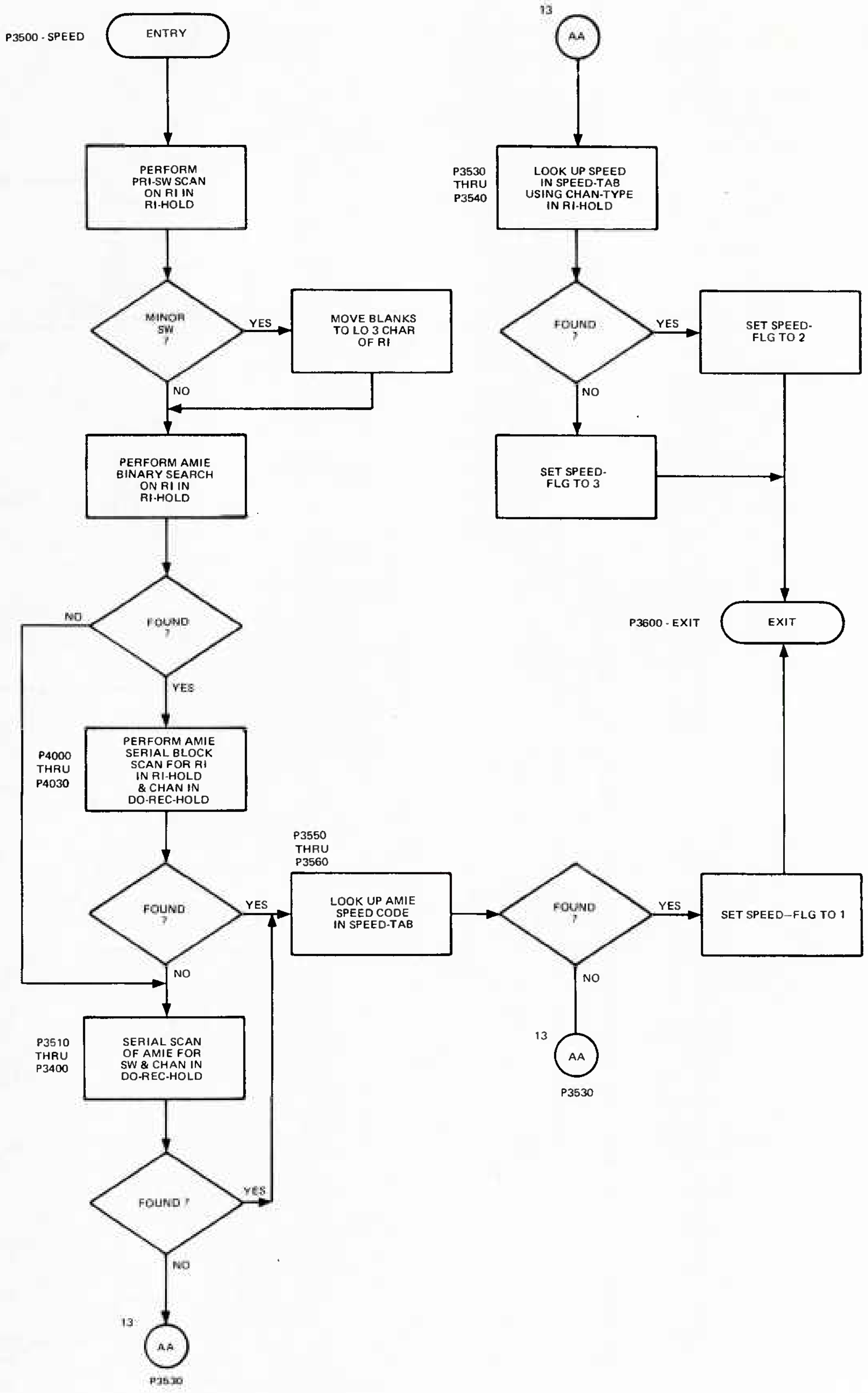
Assemble Traffic Program (Sheet 10 of 13)



Assemble Traffic Program (Sheet 11 of 13)


11
E4
P3050
COMPUTE AVG-Q-TIME
P3060
CLEAR AVG-Q-TIME COMP FIELDS
P3070
BUILD TRIB-REC & TRIB-LST-REC
P3080 THRU P3090
WRITE TRIB-REC & TRIB-LIST REC
SPEED RESOLVED ?
YES
NO
WRITE TRIB-LIST ERROR MSG
11
F4
P3100
BUILD TRIB-RPT REC
P3110 THRU P3130
WRITE TRIB-RPT REC
12
G4
P3130
12
G4
END-RUN-FLG = 1 ?
YES
12
H4
P3160
NO
CHAN-CHG-FLG = 1 ?
NO
YES
WRITE TRIB-RPT REC FROM SPACER
DETERMINE SPEED OF CURRENT RECORD
MOVE DO-REC-AREA TO DO-REC-HOLD
CLEAR DEST-COUNT ORIG-COUNT CHAN-CHG-FLG
11
A4
P3010
12
H4
P3160
CLOSE FILES
STOP

Assemble Traffic Program (Sheet 12 of 13)


P3500 - SPEED
ENTRY
PERFORM PRI-SW SCAN ON RI IN RI-HOLD
MINOR SW ?
YES
MOVE BLANKS TO LO 3 CHAR OF RI
NO
PERFORM AMIE BINARY SEARCH ON RI IN RI-HOLD
FOUND ?
NO
YES
P4000 THRU P4030
PERFORM AMIE SERIAL BLOCK SCAN FOR RI IN RI-HOLD & CHAN IN DO-REC-HOLD
FOUND ?
YES
NO
P3510 THRU P3400
SERIAL SCAN OF AMIE FOR SW & CHAN IN DO-REC-HOLD
FOUND ?
YES
NO
13
AA
P3530
P3550 THRU P3560
LOOK UP AMIE SPEED CODE IN SPEED-TAB
FOUND ?
YES
SET SPEED–FLG TO 1
NO
13
AA
P3530
13
AA
P3530 THRU P3540
LOOK UP SPEED IN SPEED-TAB USING CHAN-TYPE IN RI-HOLD
FOUND ?
YES
SET SPEED-FLG TO 2
NO
SET SPEED-FLG TO 3
P3600 - EXIT
EXIT

Assemble Traffic Program (Sheet 13 of 13)

SECTION 12 - TRAFFIC MODIFICATION AND SORT PROGRAM

12.1 PROGRAM REQUIREMENTS DESCRIPTION

12.1.1 Program Name: Traffic Modification and Sort Program

12.1.2 Program Identification: DNTRMOD1

12 1.3 Purpose

The primary function of the Traffic Modification program is to sort message segments produced by the Assemble Traffic program into the sequence required by the Discrete Message Generator and Traffic Distribution Report programs. The program eliminates all duplicate records and provides the capability to modify message characteristics, select traffic which meets specified criteria, or select traffic which is applicable to a reduced network. The traffic records retained by the program are summarized in the Originate Traffic Summary Report.

12.1.4 Requirements

The Traffic Modification and Sort program is written in ANSI COBOL and requires 200K bytes of memory when executed on the IBM 360/65.

12.2 INPUT SPECIFICATIONS

Input to the Traffic Modification and Sort program consists of the Traffic file which is produced by the Assemble Traffic program and three types of optional input cards. These cards are used to specify modification and selection criteria to the program.

12.2.1 Traffic File

Each record in the Traffic file contains a description of one message which passed through the AUTODIN system. The origin and destination of each message is specified, as are the message characteristics (line block count, security, precedence, language media format, and origin station serial number). The time of transmission (TOT), i. e. , the time at which the message entered the system, is included in the record.

The speed of service (SOS) (difference between the end of message out time at the destination switch and the TOT time) is also contained in each record on this file.

The Traffic file record format is shown in Table 11-2.

12. 2. 2 Modification Cards

The Modification cards are an optional input to the Traffic Modification and Sort program. These cards permit changes to be made to certain fields in Traffic file records.

12. 2. 2. 1 Modify Line Block Card

The Modify Line Block (MODLBC) card permits changing the line block count field to a new value when the existing value falls within a specified range. The format of this card is illustrated in Table 12-1. The card contains three values that identify the range of entries to be modified and the new entry. These should be interpreted "if the existing line block count is equal to or greater than (value 1) and is equal to or less than (value 2), change the line block count to (value 3)." A maximum of two MODLBC cards may be submitted per run.

12. 2. 2. 2 Modify Time of Transmission Card

The Modify Time of Transmission (MODTOT) card permits the message origination time to be increased by a specified number of minutes (not to exceed 1440). Only one MODTOT card may be submitted per run. The card format is illustrated in Table 12-1.

12. 2. 2. 3 Modify/Delete Tributary Card

The Modify Tributary (MODTRB) card permits changing a specified tributary name to another name. The program scans all file records and implements the change on records containing the indicated name in the origin and/or destination fields.

A maximum of 255 MODTRB cards may be submitted in one run. The Delete Tributary (DELTRB) card permits deleting all messages originated by or destined to a specified tributary. A maximum of 255 DELTRB cards may be submitted in one run. The format for these cards is illustrated in Table 12-3.

12.2.3 Network Segment Cards

To simulate a portion of the AUTODIN network the traffic must be filtered to eliminate undesired traffic. Network segmenting may be accomplished in two ways. One way is to restrict the network to only those switches, links, and tributaries which appear in the desired segment. In this case, all traffic must be deleted which does not originate and terminate at a switch in the segmented network A second, and more accurate, method is to utilize a full network and retain any traffic which originates at, terminates at, or passes through a switch in the segment portion of the network. The program provides the capability for both types of filtering.

12.2.3.1 Type 1 Network Segment Selection

The Type 1 Network Segment Selection indicates the case where the full network is simulated and the program is to retain any traffic which originates at, terminates at, or passes through a switch in the segment portion of the network. This operation is indicated by introducing a series of NETWRK cards which contain the switch names of the segmented portion of the network. The format of the NETWRK card is given in Table 12-4. A maximum of 22 NETWRK cards may be used. When this selection is indicated, the Routing and Translation files are required inputs to the program as described in Paragraph 12.2.6.

12.2.3.2 Type 2 Network Segment Selection

The Type 2 Network Segment Selection indicates the case where the network is composed of only those switches, links, and tributaries associated with the segment portion of the network. The program retains only that traffic which originates at and terminates at a switch in the segment of the network. This operation is indicated by introducing a series of NETWRK cards which contain the switch names of the segmented portion of the network. The first NETWRK card must indicate in Column 13 that Type 2 selection is desired. A maximum of 22 NETWRK cards may be used.

12.2.4 Traffic Selection Cards

Another form of traffic selection, based upon the characteristics of individual messages, is provided for in the Traffic Modification and Sort program. There are five characteristics considered in this form of selection; precedence, security, language media format, originating tributary, and destination tributary. The format of the SELECT card is illustrated in Table 12-5. It should be noted that the format of this card contains a position for each of the five characteristics but that each position need not be used. If more than one position is used on a card, this forms an "AND" condition in making the selection. For example, if Z were placed in Column 8 and T were placed in Column 10, the card would be interpreted as reading "select the message if the precedence is Flash and the security is Top Secret."

When more than one SELECT card is used, this forms an "OR" condition where a message would be selected if it met the characteristics described on any one of the cards. If a second card were added to the example, which contained Z in Column 8 and S in Column 10, these cards would be interpreted as reading "select the message if the precedence is Flash and the security is Top Secret, or if the precedence is Flash and the security is Secret."

Selection may be as general or as complex as desired by using the proper combination of SELECT cards. A maximum of 2000 cards may be used in any one program execution.

12.2.5 Combinations of MOD, NETWRK, and SELECT Cards

In one program execution, all, or any combination of the Modify, Network Segment, and Traffic Selection cards may be used. The program first makes any modifications requested, then makes a selection based upon Network Segment cards, and of the remaining records, makes a further selection based upon Traffic Selection cards.

12.2.6 Routing File and Translation File

When Type I Network Segment Selection is used, required inputs to the program are the Routing file, normally an input to the Status Generator program, and the Translation file, an output of the Status Generator program. These files are described in Section 2 and used to determine if a message passes through the segmented portion of the network.

12.3 OUTPUT SPECIFICATIONS

Program output consists of an abbreviated (optional) and sorted Traffic file, and the Originate Traffic Summary Report.

12.3.1 Sorted Traffic File

The sorted Traffic file is produced as an output by the Traffic Modification and Sort program. The record format is identical to the input Traffic file.

The sorted Traffic file contents may differ from the input file in both number of records and field contents as the result of an optional modification capability within the Traffic Modification and Sort program. The file is used as input by the Traffic Distribution Reports and the Discrete Message Generator programs.

12.3.2 Originate Traffic Summary Report

This report displays the message volume originating every hour at each primary switch contained in the sorted Traffic file. Horizontal and vertical totals show the switch originate message volume and overall message volume, by hour, respectively.

12.4 PROGRAM LOGIC

12.4.1 Program Execution

The Traffic Modification and Sort program is executed as a sequence of processes, each of which is essentially a manipulation of one of the data files which may be necessary for a particular execution of the program. If data from a particular file is not necessary or desired for a program execution, the corresponding program logic section is not executed.

12.4.1.1 Edit Modification Cards

The first section of program logic is executed only if the user enters Modification, Network Segmenting, or Traffic Selection cards with the program. If such cards are present, this section performs detailed syntax and logical editing of each card. If errors are encountered, they are noted in the program output, and execution terminates after all cards have been edited. If no errors are found, execution control passes to the next section.

12.4.1.2 Process Translation File

This section is executed only if the Type I Network Segment Selection has been specified in the appropriate NETWRK data card. The Translation file is processed by reading into memory arrays all switch and link records. Switches are flagged as to whether or not they are included in the specified NETWRK segment.

12.4.1.3 Process Routing File

This section is executed only for Type I Network Segment Selection. The Routing file, containing switch-to-switch route links, is read into a memory array. A similar array is constructed which indicates, for each switch-to-switch route in the network, whether or not at least one switch through which the route passes is included in the specified NETWRK segment.

12.4.1.4 Process Message File

This section is always executed unless errors have occurred in previous sections. Each message record is checked against each Modification, Network Segmenting, of Traffic Selection criterion which has been specified in input cards, if any. All records which pass the Network Segmenting or Traffic Selection tests, after any modifications are made, are written on a Sort Work file. If no Modification, Network Segmenting, or Traffic Selection cards were input to the job, all records are simply copied to the Sort Work file. When the last message record has been processed, the Sort Work file is sorted on the following sort keys (from major to minor in this order): Origin Switch, Time-of-Transmission, OSSN, Security, Precedence, and Destination Switch.

After the records are sorted, records which are exact duplicates on the six sort fields are purged from the file.

The resulting output file is the Sorted Traffic file (see Paragraph 12. 3. 1). The Originate Traffic Summary Report (see Paragraph 12. 3. 2) is generated from this file and printed.

Program logic flow charts are included at the end of this section.

12. 4. 2 Memory Layout

The program does not use subroutines and requires a single block of memory for executable instructions and storage arrays.

12. 5 MACHINE DEPENDENCE

The Traffic Modification and Sort program is generally machine independent. All coding conforms to ANSI Standards. However, in implementing the program on an IBM 360/65, some machine dependence is unavailable. The Identification Division describes an IBM 360/65.

Table 12-1. MODLBC Card Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-6	Card Type	MODLBC
7	Blank	
8-10	Lower Limit	Lowest Line Block Count value to be modified*
11	Blank	
12-14	Upper Limit	Highest Line Block Count value to be modified*
15	Blank	
16-18	New LBC Value	Line Block Count to be substituted*
19-80	Blank	

*This card is to be interpreted as: If the existing Line Block Count is equal to or greater than the Lower Limit and is equal to or less than the Upper Limit, change the field to New LBC Value.

Table 12-2. MODTOT Card Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-6	Card Type	MODTOT
7	Blank	
8-11	Time Increment	Value in minutes to be added to time of transmission in each record. Maximum value is 1440.
12-80	Blank	

Table 12-3. MODTRB/DELTRB Card Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-6	Card Type	MODTRB or DELTRB
7	Blank	
8-13	Old Tributary Name	Name to be replaced in origin and destination fields if MODTRB. Name to be deleted if DELTRB
14	Blank	
15-20	New Tributary Name	Name which replaces all occurrences on the Traffic File of Old Name in cc 8-13
21-80	Blank	

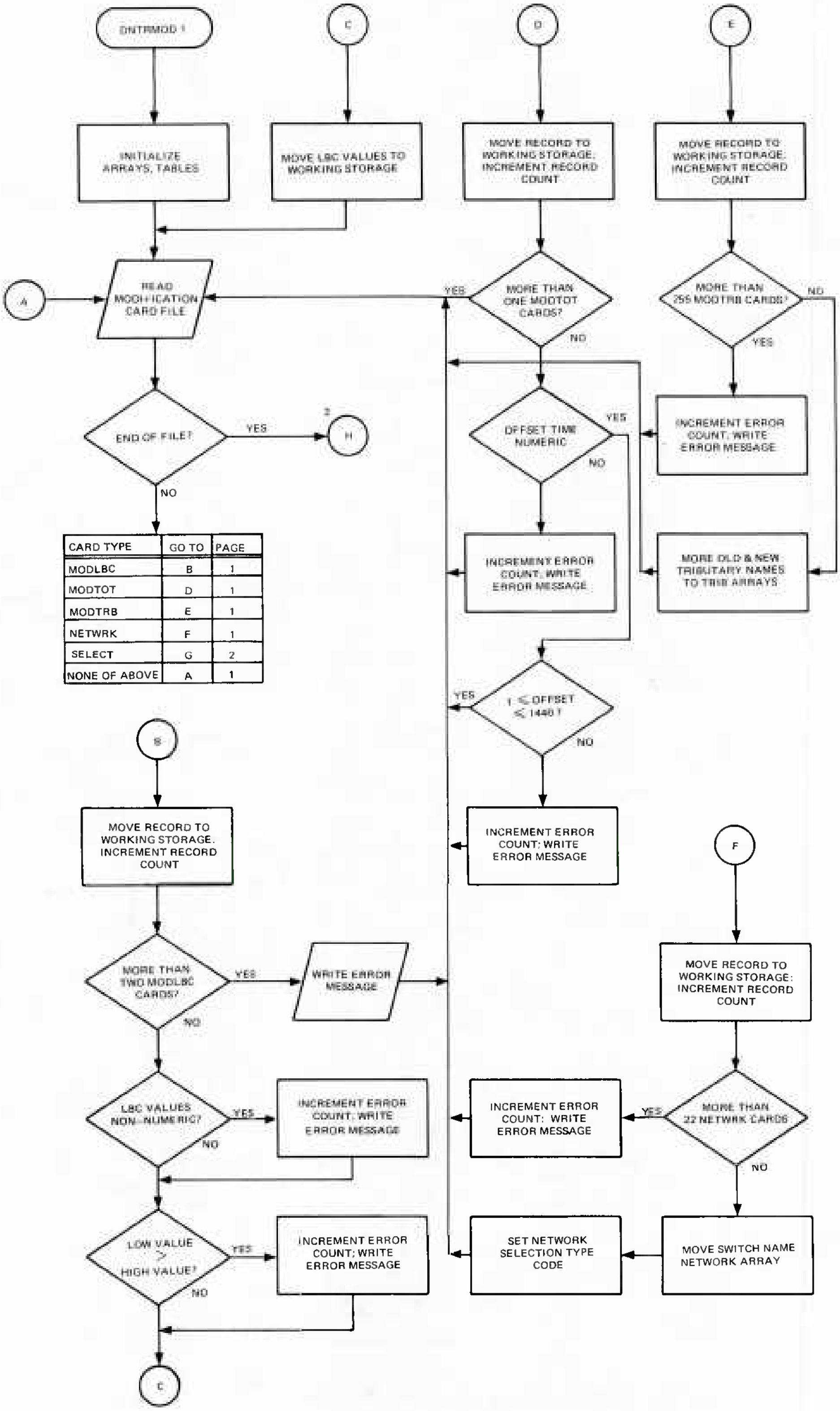
Table 12-4. NETWRK Card Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-6	Card Type	NETWRK
7	Blank	
8-11	Switch Name	Switch name to be included in simulated network
12	Blank	
13	Selection Type	If any nonblank character is present, will use Type II selection criterion (see Paragraph 14. 2. 3)*
14-80	Blank	

*This column may only be used in the first NETWRK card.

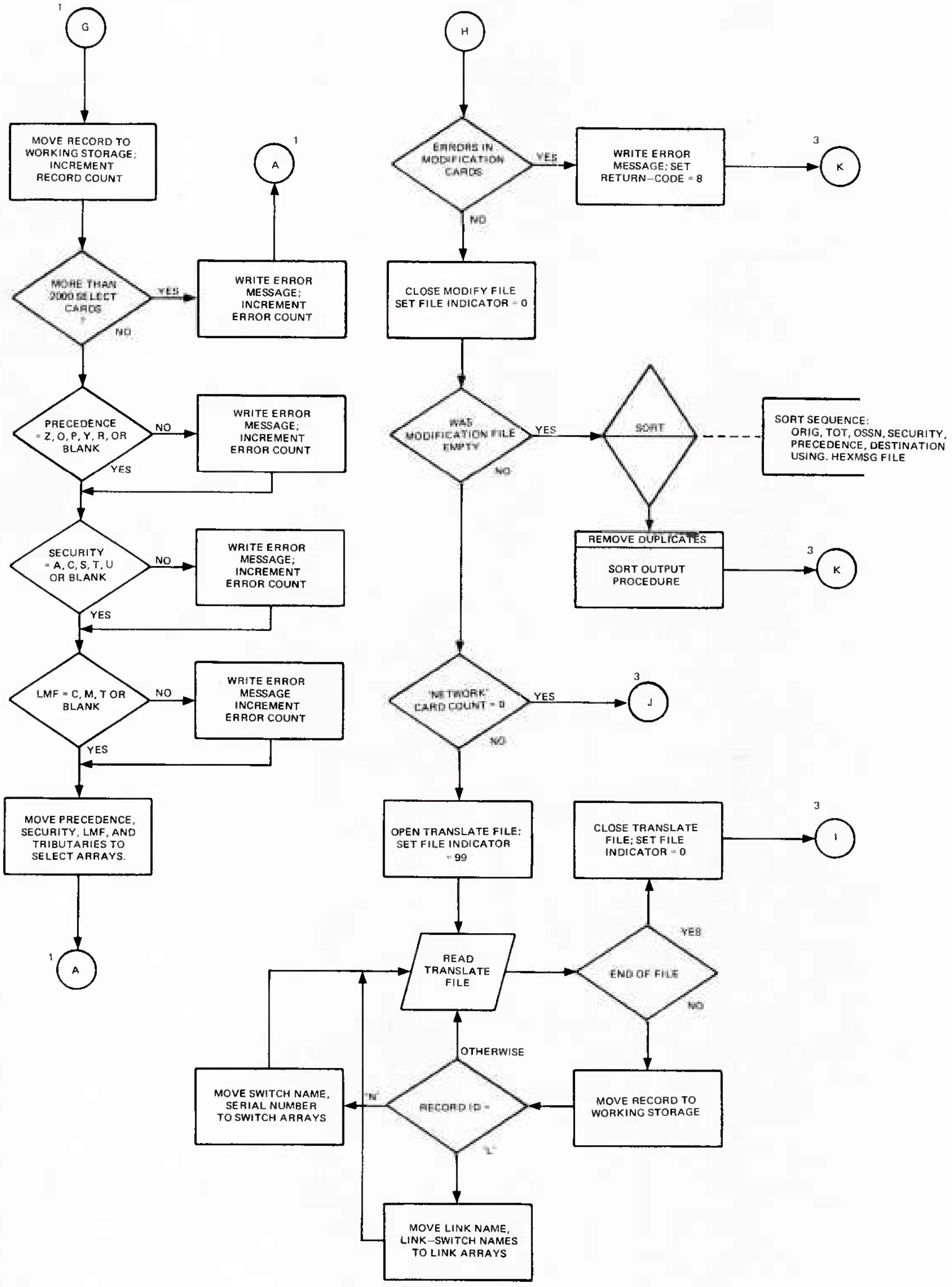
Table 12-5. Select Card Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-6	Card Type	SELECT
7	Blank	
8	Precedence	Precedence Type to be included in "AND" condition
9	Blank	
10	Security	Security type to be included in "AND" condition
11	Blank	
12	Language Media Format	LMF to be included in "AND condition
13	Blank	
14-19	Origin	Origin Tributary name to be included in "AND" condition
20	Blank	
21-26	Destination	Destination Tributary name to be included in the "AND" condition
27-80	Blank	

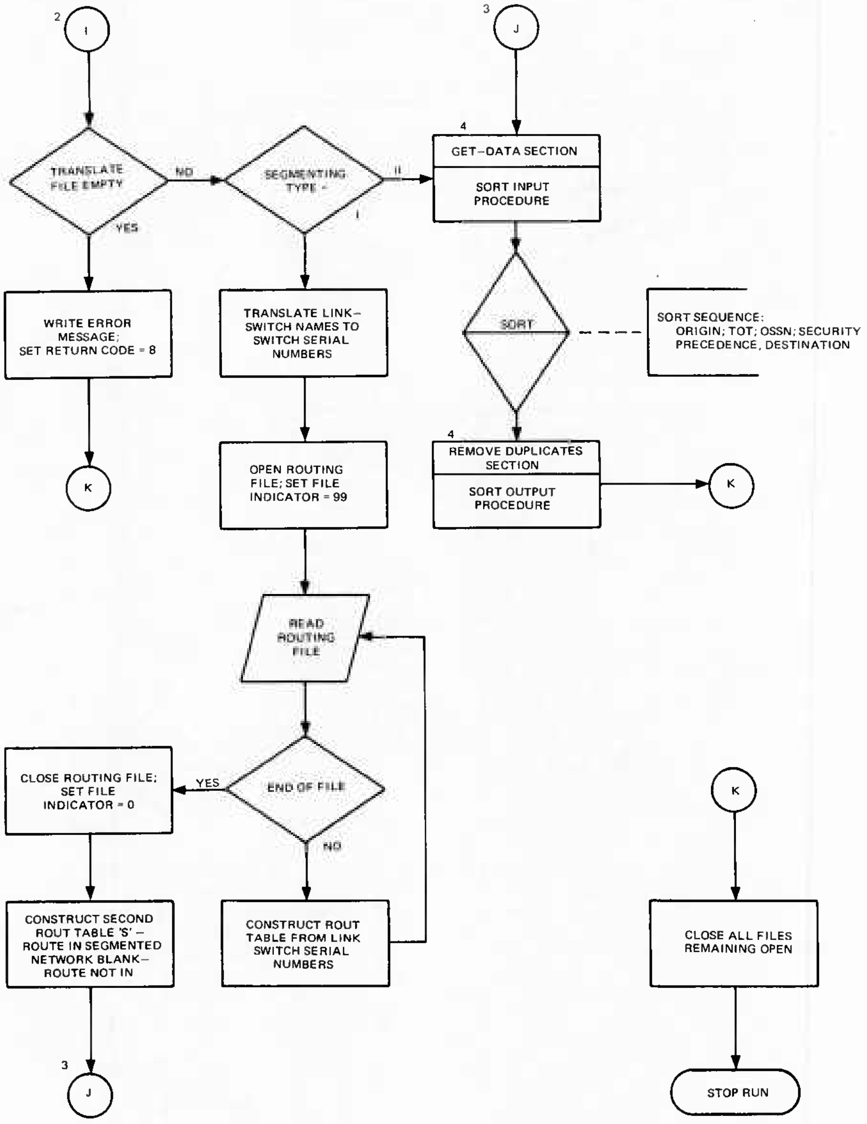

DNTRMOD 1
C
D
E
INITIALIZE ARRAYS, TABLES
MOVE LBC VALUES TO WORKING STORAGE
MOVE RECORD TO WORKING STORAGE; INCREMENT RECORD COUNT
MOVE RECORD TO WORKING STORAGE; INCREMENT RECORD COUNT
A
READ MODIFICATION CARD FILE
YES
MORE THAN ONE MODTOT CARDS?
NO
MORE THAN 255 MODTRB CARDS?
NO
YES
END OF FILE?
YES
2
H
NO
OFFSET TIME NUMERIC
YES
NO
INCREMENT ERROR COUNT; WRITE ERROR MESSAGE
INCREMENT ERROR COUNT; WRITE ERROR MESSAGE
MORE OLD & NEW TRIBUTARY NAMES TO TRIB ARRAYS
YES
1 ≤ OFFSET ≤ 1440 ?
NO
B
INCREMENT ERROR COUNT; WRITE ERROR MESSAGE
F
MOVE RECORD TO WORKING STORAGE. INCREMENT RECORD COUNT
MORE THAN TWO MODLBC CARDS?
YES
NO
WRITE ERROR MESSAGE
MOVE RECORD TO WORKING STORAGE: INCREMENT RECORD COUNT
LBC VALUES NON-NUMERIC?
YES
NO
INCREMENT ERROR COUNT; WRITE ERROR MESSAGE
INCREMENT ERROR COUNT: WRITE ERROR MESSAGE
YES
MORE THAN 22 NETWRK CARDS
NO
LOW VALUE > HIGH VALUE?
YES
NO
INCREMENT ERROR COUNT; WRITE ERROR MESSAGE
SET NETWORK SELECTION TYPE CODE
MOVE SWITCH NAME NETWORK ARRAY
C

CARD TYPE	GO TO	PAGE
MODLBC	B	1
MODTOT	D	1
MODTRB	E	1
NETWRK	F	1
SELECT	G	2
NONE OF ABOVE	A	1

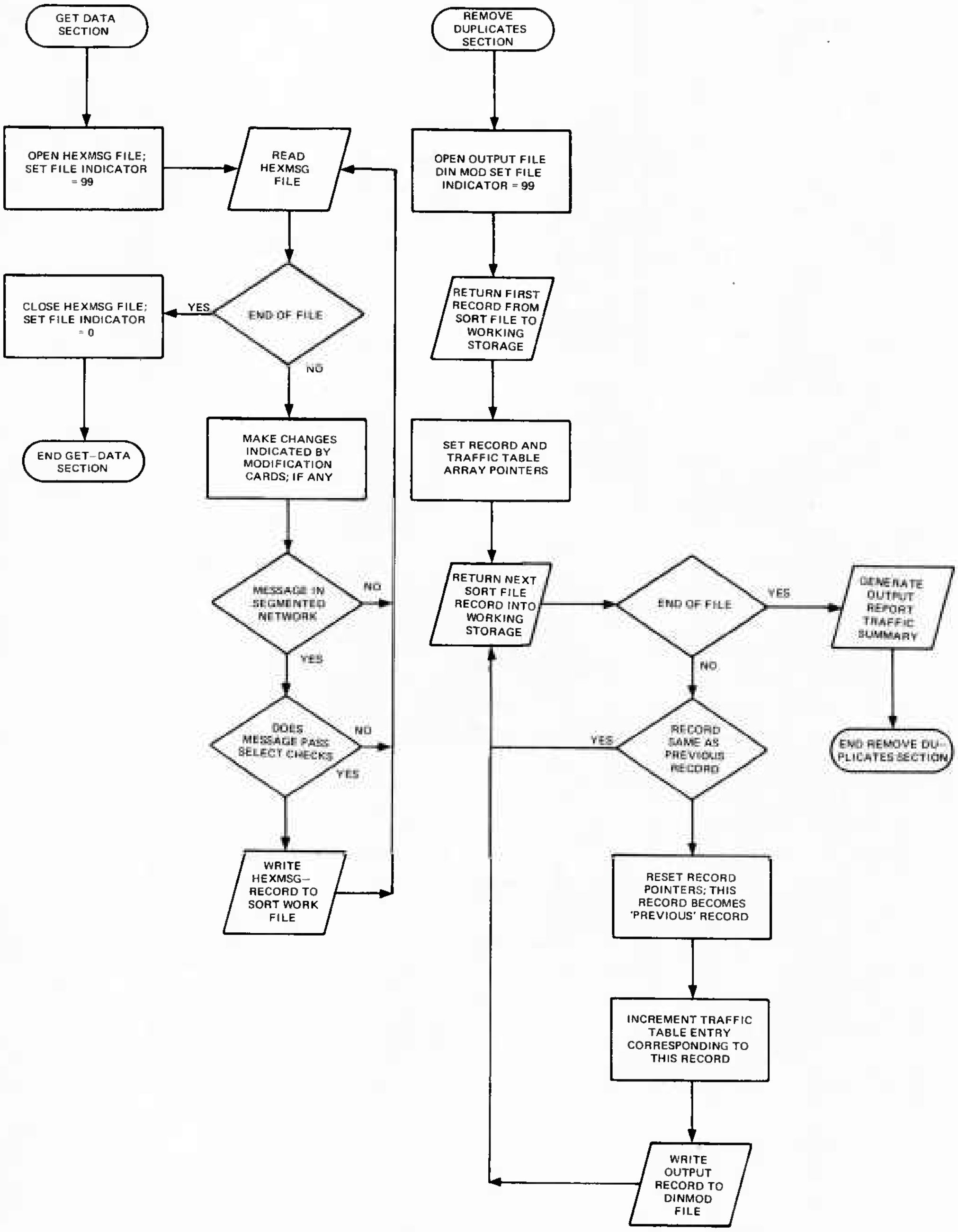
Traffic Modification and Sort Program (Sheet 1 of 4)


1
G
MOVE RECORD TO WORKING STORAGE; INCREMENT RECORD COUNT
1
A
MORE THAN 2000 SELECT CARDS ?
YES
NO
WRITE ERROR MESSAGE; INCREMENT ERROR COUNT
PRECEDENCE = Z, O, P, Y, R, OR BLANK
NO
YES
WRITE ERROR MESSAGE; INCREMENT ERROR COUNT
SECURITY = A, C, S, T, U OR BLANK
NO
YES
WRITE ERROR MESSAGE; INCREMENT ERROR COUNT
LMF = C, M, T OR BLANK
NO
YES
WRITE ERROR MESSAGE INCREMENT ERROR COUNT
MOVE PRECEDENCE, SECURITY, LMF, AND TRIBUTARIES TO SELECT ARRAYS.
1
A
H
ERRORS IN MODIFICATION CARDS
YES
NO
WRITE ERROR MESSAGE; SET RETURN-CODE = 8
3
K
CLOSE MODIFY FILE SET FILE INDICATOR = 0
WAS MODIFICATION FILE EMPTY
YES
NO
SORT
SORT SEQUENCE: ORIG, TOT, OSSN, SECURITY, PRECEDENCE, DESTINATION USING. HEXMSG FILE
REMOVE DUPLICATES
SORT OUTPUT PROCEDURE
3
K
'NETWORK' CARD COUNT = 0
YES
NO
3
J
OPEN TRANSLATE FILE; SET FILE INDICATOR = 99
CLOSE TRANSLATE FILE; SET FILE INDICATOR = 0
3
I
READ TRANSLATE FILE
END OF FILE
YES
NO
OTHERWISE
MOVE SWITCH NAME, SERIAL NUMBER TO SWITCH ARRAYS
'N'
RECORD ID =
'L'
MOVE RECORD TO WORKING STORAGE
MOVE LINK NAME, LINK-SWITCH NAMES TO LINK ARRAYS

Traffic Modificationand Sort Program (Sheet 2 of 4)


2
I
TRANSLATE FILE EMPTY
NO
YES
SEGMENTING TYPE =
II
I
3
J
4
GET–DATA SECTION
SORT INPUT PROCEDURE
SORT
SORT SEQUENCE:
ORIGIN; TOT; OSSN; SECURITY PRECEDENCE, DESTINATION
WRITE ERROR MESSAGE;
SET RETURN CODE = 8
TRANSLATE LINK–SWITCH NAMES TO SWITCH SERIAL NUMBERS
K
OPEN ROUTING FILE; SET FILE INDICATOR = 99
4
REMOVE DUPLICATES SECTION
SORT OUTPUT PROCEDURE
K
READ ROUTING FILE
CLOSE ROUTING FILE; SET FILE INDICATOR = 0
YES
END OF FILE
NO
K
CONSTRUCT SECOND ROUT TABLE 'S' – ROUTE IN SEGMENTED NETWORK BLANK– ROUTE NOT IN
CONSTRUCT ROUT TABLE FROM LINK SWITCH SERIAL NUMBERS
CLOSE ALL FILES REMAINING OPEN
3
J
STOP RUN

Traffic Modification and Sort Program (Sheet 3 of 4)


GET DATA SECTION
OPEN HEXMSG FILE; SET FILE INDICATOR = 99
READ HEXMSG FILE
END OF FILE
YES
NO
CLOSE HEXMSG FILE; SET FILE INDICATOR = 0
END GET–DATA SECTION
MAKE CHANGES INDICATED BY MODIFICATION CARDS; IF ANY
MESSAGE IN SEGMENTED NETWORK
NO
YES
DOES MESSAGE PASS SELECT CHECKS
NO
YES
WRITE HEXMSG–RECORD TO SORT WORK FILE
REMOVE DUPLICATES SECTION
OPEN OUTPUT FILE DIN MOD SET FILE INDICATOR = 99
RETURN FIRST RECORD FROM SORT FILE TO WORKING STORAGE
SET RECORD AND TRAFFIC TABLE ARRAY POINTERS
RETURN NEXT SORT FILE RECORD INTO WORKING STORAGE
END OF FILE
YES
NO
GENERATE OUTPUT REPORT TRAFFIC SUMMARY
END REMOVE DU– PLICATES SECTION
RECORD SAME AS PREVIOUS RECORD
YES
RESET RECORD POINTERS; THIS RECORD BECOMES 'PREVIOUS' RECORD
INCREMENT TRAFFIC TABLE ENTRY CORRESPONDING TO THIS RECORD
WRITE OUTPUT RECORD TO DINMOD FILE

Traffic Modification and Sort Program (Sheet 4 of 4)

SECTION 13 - TRAFFIC DISTRIBUTION REPORT PROGRAM

13.1 PROGRAM REQUIREMENT DESCRIPTION

13.1.1 Program Name: Traffic Distribution Report Program

13.1.2 Program Identification: DNTRFDR1

13.1.3 Purpose

This program provides a summary of the AUTODIN traffic which was assembled from the sample data. The summary permits a judgment to be made concerning the validity of this traffic sample. The report also indicates the speed of service (SOS) attained in the network as computed from the traffic sample data.

13.1.4 Requirements

The program is written in generalized FORTRAN IV language and requires a minimum of 60K bytes or 15K 32-bit words for execution on an IBM 360 System.

13.2 INPUT SPECIFICATIONS

Input to the program is one disk file and an optional set of input parameter cards.

13.2.1 Sorted Traffic Input File

The input file to the program is the Sorted Traffic file generated by the Traffic Modification and Sort program. Creation of this file is described in Paragraph 12.3.

13.2.2 Input Parameter Cards

The optional input parameter cards permit specifying a RADAY for the title line of the printout. If it is desired that the summary not include all switches in the network, a list of those switches to be included may be inserted. The format of these cards is given in Tables 13-1 and 13-2.

13.3 OUTPUT SPECIFICATIONS

The program output is a switch by switch listing of the volume of traffic generated by the tributaries at a given switch and the distribution of this traffic to other switches in the network.

13.4 PROGRAM LOGIC

13.4.1 RADAY Card

If the first input card read by the program specifies the word RADAY in Columns 1-5, the value specified in Columns 7-9 is used in the report heading to indicate the RADAY that the sample was taken. The word IBIT in Columns 11-14 sets a switch for later use as described in Paragraph 13.4.3.

13.4.2 Switch Cards

The program contains a fixed list of the names of the switches in AUTODIN. This list is used in preparing the summary report unless a new list is introduced. If any cards follow the RADAY card, it is assumed that they are Switch cards and the fixed list is replaced.

13.4.3 Report Preparation

The first three characters of the tributary identification used in the traffic assembling process are the same as the second, third, and fourth letters of a switch identification. Thus, it can be determined in each of the traffic records the source and destination switches. When the first record is read, the source and destination are tested against the list of switches. If either is not in the list, the next record is read. When a valid record is found, the summing matrix is initialized, the first record is counted, and the identifying characteristics of the message are saved.

As each successive record is read, the first test made is for a change of originating switch. When this occurs, the summary for the previous switch is written. The second test made is for a multiple address message; i.e., are the message characteristics the same as the previous message? If the record indicates a new message. the originated message counters are increased, the destination determined, and the appropriate counters increased. If the message is multiple-addressed, the IBIT option was not specified, and the new delivery is going to the originating switch, the destination counters are increased. If the IBIT option was specified, or if the destination is other than the originating switch, the destination of the current record is compared to the destination of the previous record. If they are not the same, all destination counts are increased. If they are the same, only the SOS counters are increased.

When a change of originating switch is detected or when the end of the input file is reached, the average SOS to each switch is computed, and this figure and the other values in the summing matrix are printed.

Program logic flow charts are included at the end of this section.

13.5 MACHINE DEPENDENCE

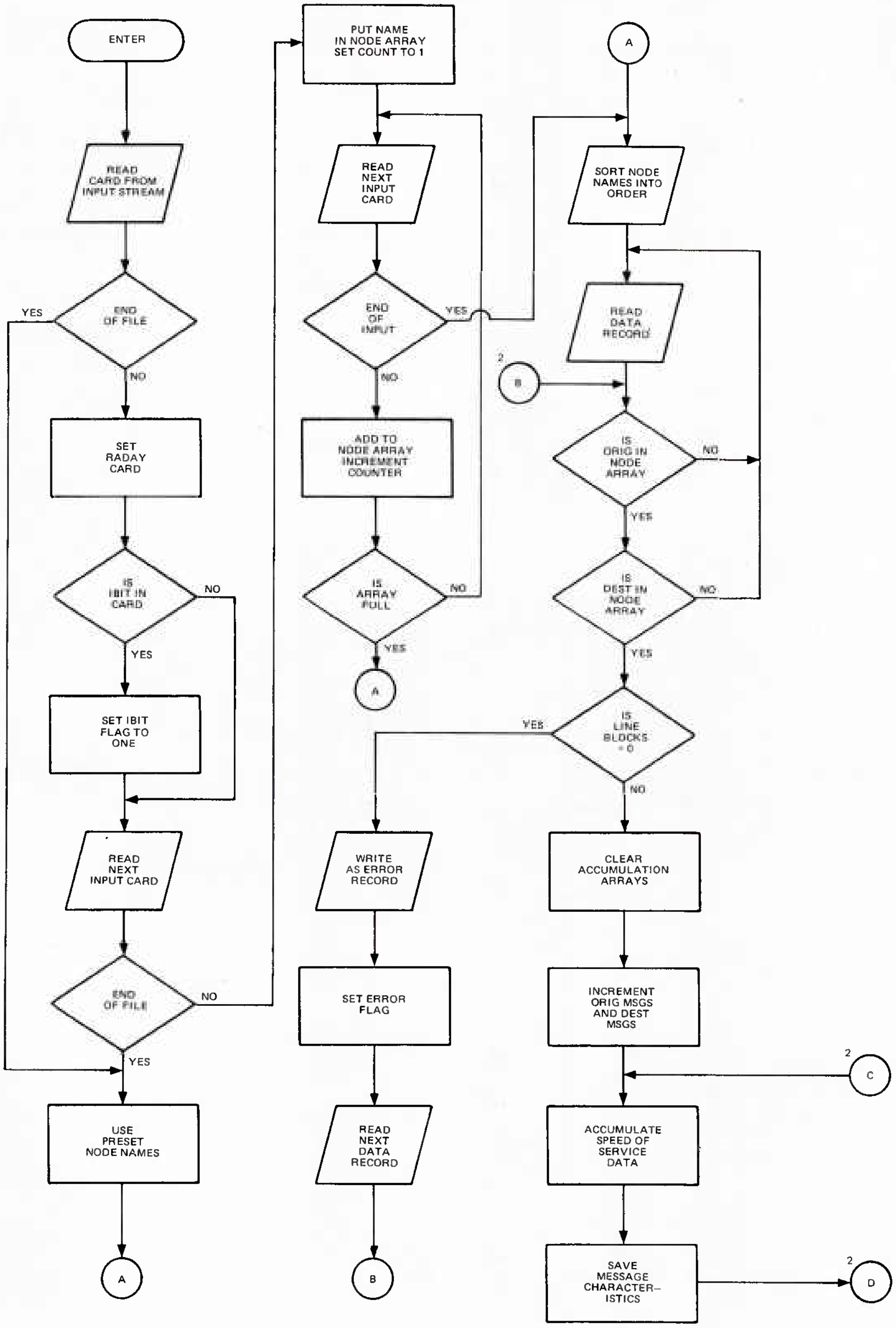
The program is written in FORTRAN IV language and is essentially machine independent.

Table 13-1. RADAY Card Format

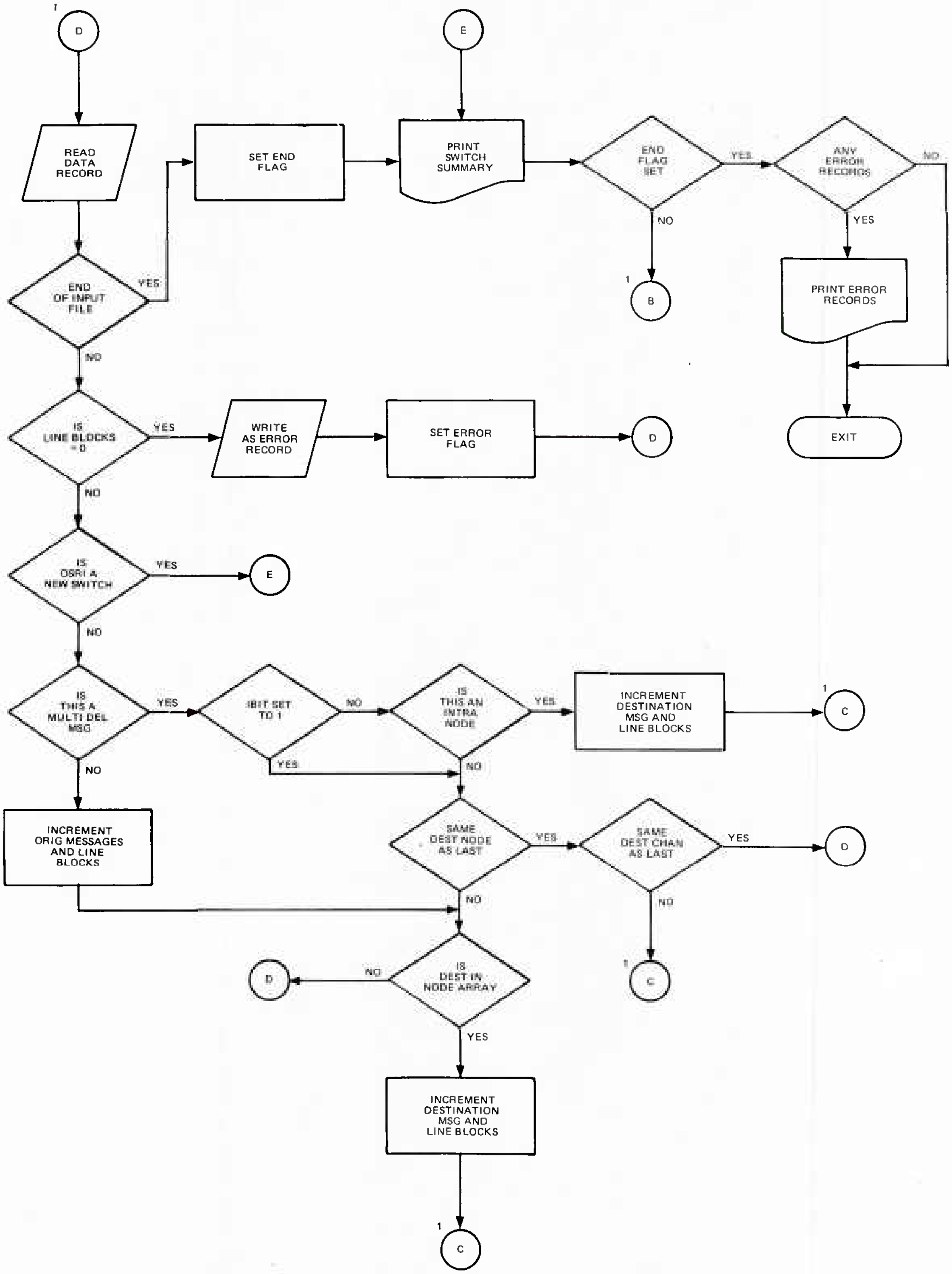
CARD COLUMN	DESCRIPTION	EXPLANATION
1-5	Identifier	RADAY
6	Blank	
7-9	Sample Date	May contain any three-character identifier.
10	Blank	
11-14	Intra-Switch Flag	Contains IBIT if intra-switch messages are to be counted in the same manner as interswitch messages.

Table 13-2. Switch Card Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-4	Switch Identifier	Contains any valid AUTODIN switch identifier (RUAD, RUAO, etc.).
5-80	Blank	



Traffic Distribution Report Program (Sheet 1 of 2)



Traffic Distribution Report Program (Sheet 2 of 2)

SECTION 14 - LINK LOAD PROGRAM

14.1 PROGRAM REQUIREMENT DESCRIPTION

14.1.1 Program Name: Link Load Program

14.1.2 Program Identification: DNLNKLD1

14.1.3 Purpose

The Link Load program creates a file which defines a network link configuration based on an examination of link traffic which actually occurred during a sample period. One record is written on the Network Link file to define the number of equal-speed, two-way channels, which existed between each network switch pair. The program uses the AMIE Link file to obtain an up-to-date and consistent set of characteristics related to each link.

Concurrent with this processing, three reports are generated. One, the AUTODIN Inter-Switch Link Load Report, presents the actual message volume, by priority, that passed over each link output channel. The second, the Channel Activity Report, defines channel inactivity periods and probable channel outage periods on a minute-by-minute basis for a 24-hour period. A third report lists the Network Link file and prints warning messages for certain data conditions which require manual intervention.

14.1.4 Requirements

The Link Load program is written in ANSI COBOL and requires 114K bytes of core when executed on an IBM 360/65.

14.2 PROGRAM INPUTS

There are three inputs to the program: switch-RADAY cards, the AMIE Extract Link file, and the Header Extract Link file.

14.2.1 Switch-RADAY Card Set

The switch-RADAY card set communicates to the Link Load program the actual RADAY for which traffic is supplied for a given simulation. It is possible that the

RADAY for data submitted by one or more switches could be for other than the standard RADAY. The program uses the card set to ensure that only link traffic for a specified day for each switch is used to establish the network link configuration.

The standard RADAY for the simulation is identified on the first card of the switch-RADAY card set. One card for each switch in the network follows the standard RADAY card. These cards identify the sample day for the indicated switch. The card sequence following the standard RADAY card may be in any order. The format for the standard and following RADAY cards is illustrated in Table 14-1.

14.2.2 AMIE Extract Link File

The AMIE Extract Link file, produced as an output by the AMIE Extract program, is read into an internal table by the Link Load program. The AMIE link data is used to process link traffic records and to ensure a consistent source of link characteristics when generating the network link configuration.

The file is disk resident and sequentially organized with 80-character (EBCDIC) logical records and 100 logical records per block. The record format is illustrated in Table 10-4.

14.2.3 Header Extract Link File

The Header Extract Link file is produced as an output by the Header Extract program. Each record on this file describes the characteristics of a message which passed across a link. The program examines the record in terms of RADAY and, if found acceptable, enters applicable data from the record into an internal program table. This table provides the program with the information required to reconstruct the network link configuration after Link file processing is complete.

The file is organized sequentially on tape with 29-character logical records and 150 logical records per block. The record format is illustrated in Table 9-5.

14.3 PROGRAM OUTPUTS

Program output is composed of the Network Link file, the Link File Listing, the AUTODIN Interswitch Link Load Report, and the Channel Activity Report.

14.3.1 Network Link File

The Network Link file contains one record defining the number of equal-speed, two-way channels existing between each network switch pair during the sample period. The file is created by the program through an examination of the link traffic which actually occurred during the period of interest. The AMIE Link file is used by the Link Load program to determine the destination switch of each link, and to provide a source of consistent information about the link (speed, Line Traffic Controller number, and maximum security capability). The program computes the average queue delay in seconds at each end of a link, and inserts this information in each output record.

Under certain data conditions, such as an unlisted link in AMIE, the program may be unable to determine appropriate values for an output Link file record. In these cases, the field is filled with one right-justified asterisk and spaces. An appropriate message appears in the Link File Listing when this occurs. It is mandatory that these records be corrected prior to using the data in succeeding programs.

The record format for the Network Link File is illustrated in Table 14-2.

14.3.2 Network Link File Listing

Each line on the Network File Listing represents one record in the Network Link file. A sequence number identifies the position of the record in the file. All warning messages on the listing must be examined and acted upon to ensure successful execution of succeeding simulator programs which use the Network Link file. Warning messages and appropriate corrective action are described in Paragraph 15.5 of the User's Manual.

14.3.3 AUTODIN Interswitch Link Load Report

The AUTODIN Interswitch Link Load Report presents a one-page link traffic summary for each switch included in the sample. The numeric value appearing on the second line of each report page identifies the sample day for the specified switch. The RADAY value appearing on the first line of the report represents the actual simulation standard RADAY.

The report lists the total number of messages and the line block count, by priority, that originated from each channel emanating from the switch. Horizontal and vertical totals provide summary information by channel and switch, respectively.

The link traffic which passes over a link channel, but is not defined in the AMIE Link file, is flagged in the report with an asterisk under the destination switch column.

14.3.4 Channel Activity Report

Each page of this report displays two 24-hour arrays showing channel activity on a minute-by-minute basis. The first array on each page shows channel inactivity periods (no traffic across link) by means of asterisks in the appropriate array minute cells. The second array shows probable channel outage periods (traffic in queue for a specified channel, but no traffic starting out of switch on the channel during the indicated interval) by means of asterisks.

An asterisk will appear under DEST SWITCH when the channel is not presented in the AMIE Link file. The numeric value appearing next to RADAY on line two of each page of the report represents the data sample day for the specified switch.

14.4 PROGRAM LOGIC

14.4.1 Program Execution

The primary function of the program is to examine all message header records representing traffic which passed across a link during the period of interest and thereby reconstruct the network link configuration. In the process of this examination, the program produces several summary reports describing link activity. The program logic which accomplishes this consists of six segments and three primary internal tables. The tables and their functions are as follows:

1. SW-RADAY-TAB - This table contains the switch-RADAY card set which is supplied as parametric input to the program. There is one record created in the table for each primary switch in the card set. Each record contains the RADAY and value corresponding to the day prior to the RADAY.

During processing of the AMIE Link and Header Extract Link files two internal tables are created: the AMIE-TAB and the NODE-TAB. Pointers to the starting position of each primary switch block in each of these tables are entered into the appropriate SW-RADAY-TAB primary switch entries. The size of each set of primary switch blocks is also retained in the corresponding SW-RADAY-TAB record.

2. AMIE-TAB - This internal program table holds all records contained on the AMIE Link file. The table is in sort order by origin switch and channel. Pointers to the starting position of each primary switch block, as well as the block size, are stored in the corresponding primary switch SW-RADAY-TAB record. The table provides a consistent source of link characteristics.

3. NODE-TAB - Records in this table are built from an analysis of each link message record contained on the Header Extract Link file. Each NODE-TAB record represents a one-way path between two switches and contains a complete description of the path as obtained from the AMIE-TAB. A count of the number of messages traversing the path and the total queue time at the path origin switch (for low priority messages) are also contained in each NODE-TAB entry. The table is in sort order by origin switch and channel. Pointers to the starting position of each primary switch block, as well as the block size, are stored in the corresponding primary switch SW-RADAY-TAB record. The table is used by the program to reconstruct the network link configuration for the sample period.

The program logic executes sequentially in the following six functional areas:

1. The switch-RADAY card set is input, validated, and loaded into the SW-RADAY-TAB. The Julian date day value for the day prior to each specified RADAY is computed and entered into the table.

2. Each record in the AMIE Link file is read, validated, and loaded into the AMIE-LINK-TAB. A pointer to the starting position of each primary

switch block and the respective table block size are placed in the corresponding switch record in the SW-RADAY-TAB.

3. As part of a COBOL Internal Sort Input Procedure, each Header Extract Link record is read, validated, and selected by RADAY criteria. If the SOMI (day) field corresponds to the specified SW-RADAY-TAB RADAY entry for the reporting switch, the SOMI and SOMO time fields are converted to minutes and the record is released to the internal sort file. If the record is rejected by this RADAY test it is accepted if the SOMI field corresponds to the day prior to the RADAY and the SOMO field equals the RADAY value specified in the SW-RADAY-TAB. This latter test provides for acceptance of message data which enters a switch prior to the RADAY but exits from the switch during the RADAY. Conversion of the time fields to minutes is accomplished to facilitate the preparation of the Channel Activity Report in the COBOL Internal Sort Output Procedure. Those time values which do not equal the specified RADAY are converted to the dummy value of 9999.

4. The internal Sort file, which contains all acceptable link records, is sorted into origin switch and channel number sequence.

5. A COBOL Sort Output Procedure is used to create the internal NODE-TAB from link records returned from the Sort and from data contained in the AMIE-TAB. One entry is made in the NODE-TAB for each unique one-way path between two switches. The total number of messages traversing the path and the total queue time for low priority messages are retained in each NODE-TAB record. Simultaneous with the generation of the NODE-TAB, the Interswitch Link Load Report and Channel Activity Report are created.

   The Interswitch Link Load Report is computed by retaining counts by priority for each channel within a switch. Since the input record sort sequence is in the required report order, line totals are printed by channel, rolled into switch totals, and cleared.

The Channel Activity Report consists of a one-page report for each channel. Two arrays are printed on each page. These arrays are based on two internal tables, each of which contain 1440 one-character positions. Each position sequentially represents 1 minute during a 24-hour period. One table (TIME-SEQ-TAB-IN) contains an asterisk in each position representing the time period when one or more messages came into the switch destined for the channel. A second table (TIME-SEQ-TIME-OUT) contains an asterisk in each position representing the time period when one or more messages left the switch on the specified channel.

The first array on each page of the Channel Activity Report shows channel inactivity periods (no traffic across link) by means of asterisks in the appropriate array minute cells. This array is essentially the inverse of the internal TIME-SEQ-TAB-OUT. The second printed array shows probable channel outage periods (traffic in queue for time period but no traffic starting out of switch on the channel during the indicated interval). This array is based on a minute-by-minute comparison of the TIME-SEQ-TAB-IN and TIME-SEQ-TAB-OUT fields. A probable outage occurs when, for corresponding minute fields, the former table contains an asterisk and latter is blank.

6. The final segment of the program creates the Network Link file from the NODE-TAB and produces a sequenced listing of the file.

   The NODE-TAB record contains one entry for each unique one-way path between switches as determined from an evaluation of the link message data. The primary function of the logic in this section of the program is to examine the entries in the NODE-TAB and determine how many two-way, equal-speed paths (i. e. links) exist between each node.

   The table is sequentially scanned, one primary switch block at a time (pointers are obtained in RADAY-SW-TAB). An intermediate table, CONSTRUC-LNK-TAB, is completed for each new link encountered in

NODE-TAB. Up to eight links, corresponding to the eight possible circuit speeds between two switches, may be built at one time within this table. All applicable origin-to-destination paths are first entered into this table. The corresponding destination-to-orgin block of NODE-TAB is then scanned for all opposite direction paths between the two switches. Each record in the CONSTRUC-LNK-TAB should contain an equal number of bi-directional, equal-speed paths following origin-to-destination and destination-to-origin NODE-TAB evaluation. Those records in CONSTRUC-LNK-TAB metting this criteria are used to create link records for output to the Network Link file. Those records which do not correctly define link records are also written on the Network Link file, but contain asterisks in unresolved fields. The link records are listed in the Network Link File Listing with appropriate messages explaining any fields which are not resolved.

14.4.2 Internal Primary Switch Table Modification

The program contains an internal table (SW-RADAY-VAL) which contains a list of all valid AUTODIN primary switches. The table is capable of containing 24 switch values, but is currently initialized with 21 switches. If the table is modified, the field SW-COUNT-PLUS1 must be set to the number of values in the table plus one. Switches may appear in any order within the table.

14.4.3 Error Diagnostics and Messages

1. ORIG-SW IN ERROR IN AMIE LINK-REC (ORIG, DEST)

   A validity check of the origin switch in an AMIE Link file record indicated that a nonprimary switch designator filled this field. The program bypasses this record and continues processing the AMIE Link file. If an excessive number of errors of this type are present, an update of the AMIE Link file is indicated. The impact of this error on program execution is an increase in the number of incomplete records in the Network Link file on output (i.e., an increase in the number of manual modifications which are required on this file).

2. AMIE FILE LENGTH GT 400 RECS

   The AMIE Link file contains a record count greater than 400 records. Since the program is designed to terminate execution if over 400 records are present, the job must be rerun. Prior to resubmission of the job, decrease the AMIE Link file size or increase the Link Load program AMIE Link Table size.

3. AMIE FILE CONTAINS 0 RECORDS

   The program terminated execution as a result of finding zero records on the AMIE Link file. Resubmit the job with an AMIE Link file that contains at least one record, but less that 401 records.

4. NR HEXLINK SW ERRORS (COUNT)

   This message provides a count of the number of Header Extract Link records which were discarded because the origin switch field contained an invalid primary switch value. No action is required unless the count is large. In such a case an update to the Link Load program primary switch table might be required by a maintenance programmer.

5. NR HEXLINK TIME DISCARDS (COUNT)

   This message provides a count of the number of Header Extract Link records which were discarded because they represent messages which did not pass through AUTODIN on the sample RADAY. If this count is excessive it is possible that a switch RADAY card contains an erroneous RADAY value. Refer to the Header Extract program summary printout for evaluation of the count.

6. SORT CONTAINS 0 RECORDS WITH ID LISTED IN SW-LST

The program found zero acceptable records in the Header Extract Link file. It is probable that the switch RADAY card set is erroneous or the wrong Header Extract Link file was used for input. Correct the appropriate error and resubmit the job.

7. DEST-SW-IN LINK OUTPUT REC INVALID (DEST, ORIG)

The program determined that an invalid destination switch was present in a Header Extract Link File record. If the message appears a significant number of times, the job should be resubmitted following appropriate corrective action. It is probable that the primary switch table in either the Header Extract or Link Load programs should be updated. The program continues execution following this error condition, but the Network Link file contains an erroneous destination switch and lack of related switch data. The proper switch destination value should be entered into the corresponding Network Link file output record.

8. NODE TAB REC CONTAINS INVAL SPEED CODE (SPEED, ORIG, DEST)

The program found an invalid speed code in an internal switch table entry during creation of a record for the Network Link file. If this error occurs, it is probable that the internal speed code table of the Link Load program requires updating. Following appropriate action by a maintenance programmer, the job should be rerun.

9. DEST-SW-IN LINK OUTPUT REC NOT IN NODE TAB (ORIG, DEST)

The program determined that zero link messages passed from the link destination to the link origin during the sample period. No action is required as the result of this message since an equivalent warning message will appear in the Network Link File Listing.

Program logic flow charts are included at the end of this section.

14.4.4 Memory Layout

The program does not use subroutines and requires a single block of memory for executable instructions and storage arrays.

14.5 MACHINE DEPENDENCE

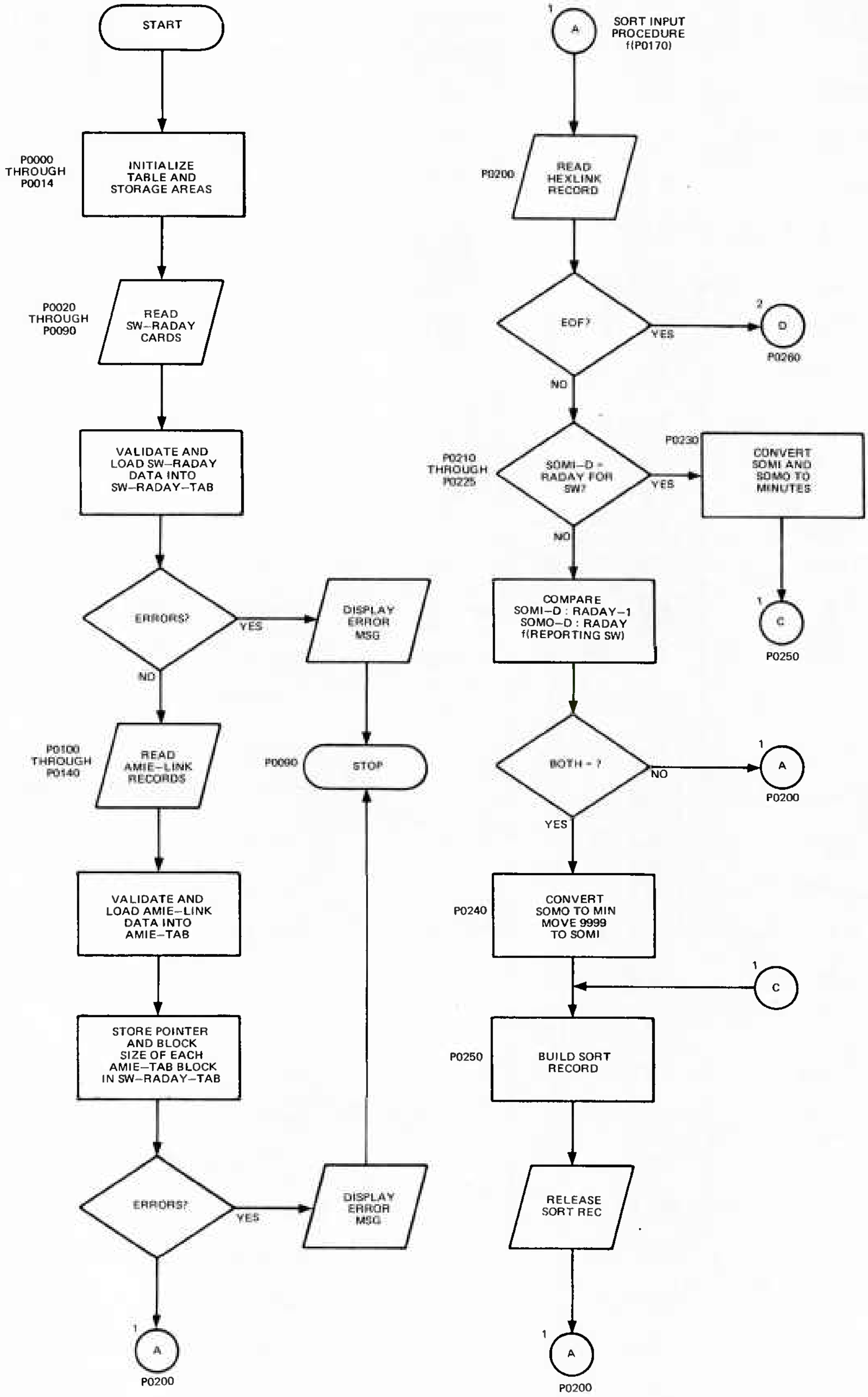
The Link Load program is generally machine independent. All coding conforms to ANSI COBOL standards. However, in implementing the program on an IBM 360/65 some machine dependence was unavoidable. The Identification Division describes an IBM 360/65.

Table 14-1. Standard RADAY and Switch-RADAY Card Formats

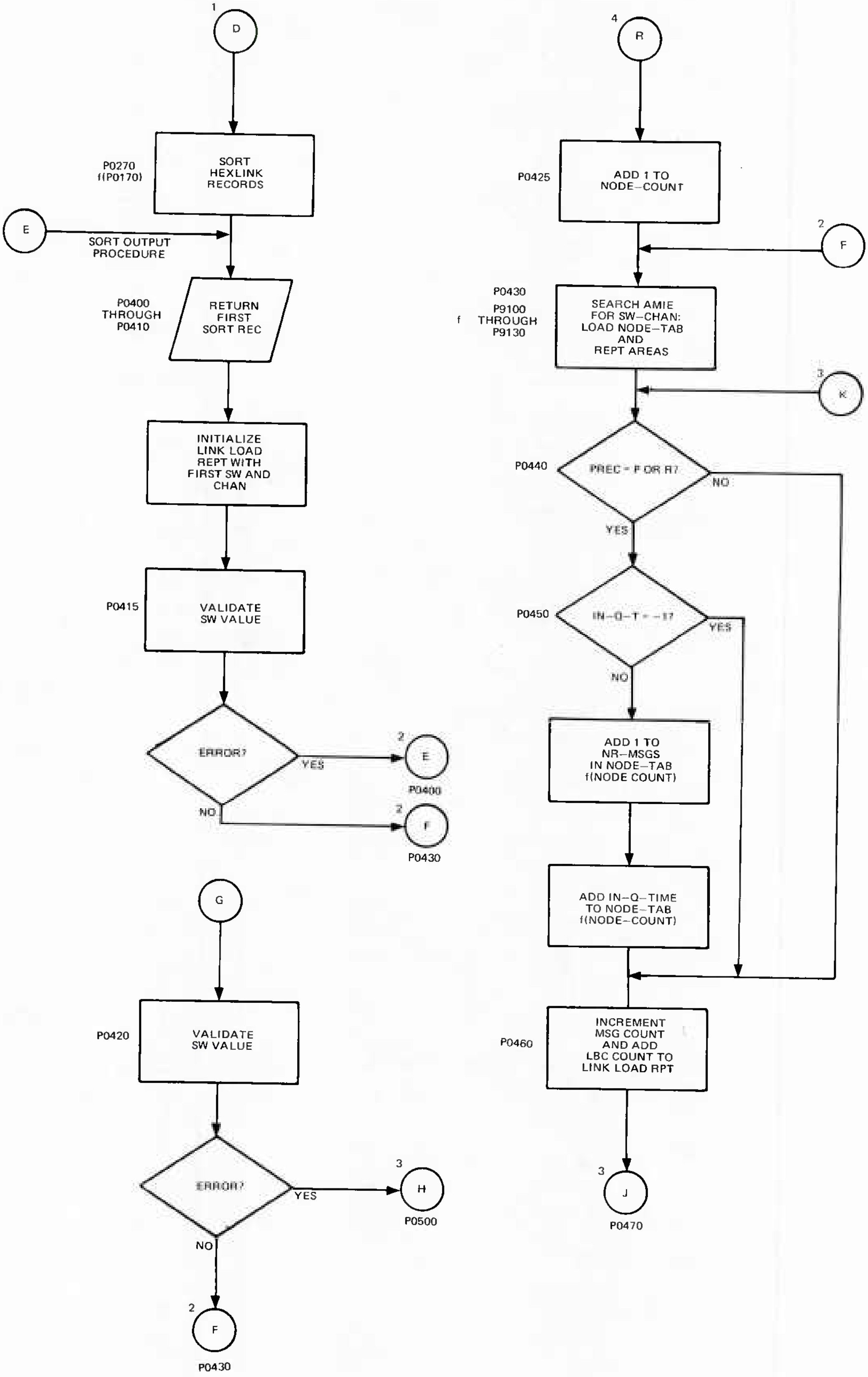
CARD COLUMN	DESCRIPTION	EXPLANATION
STANDARD RADAY CARD FORMAT		
1 - 5	Blank	
6 - 8	Standard RADAY	Numeric Julian Day between 001 and 366
9 - 80	Blank	
SWITCH-RADAY CARD FORMAT		
1 - 4	Primary Switch	
5	Blank	
6 - 8	RADAY for sample submitted for switch in cc 1 - 4	Numeric Julian Day between 001 and 366
9 - 80	Blank	

Table 14-2. Network Link File Format

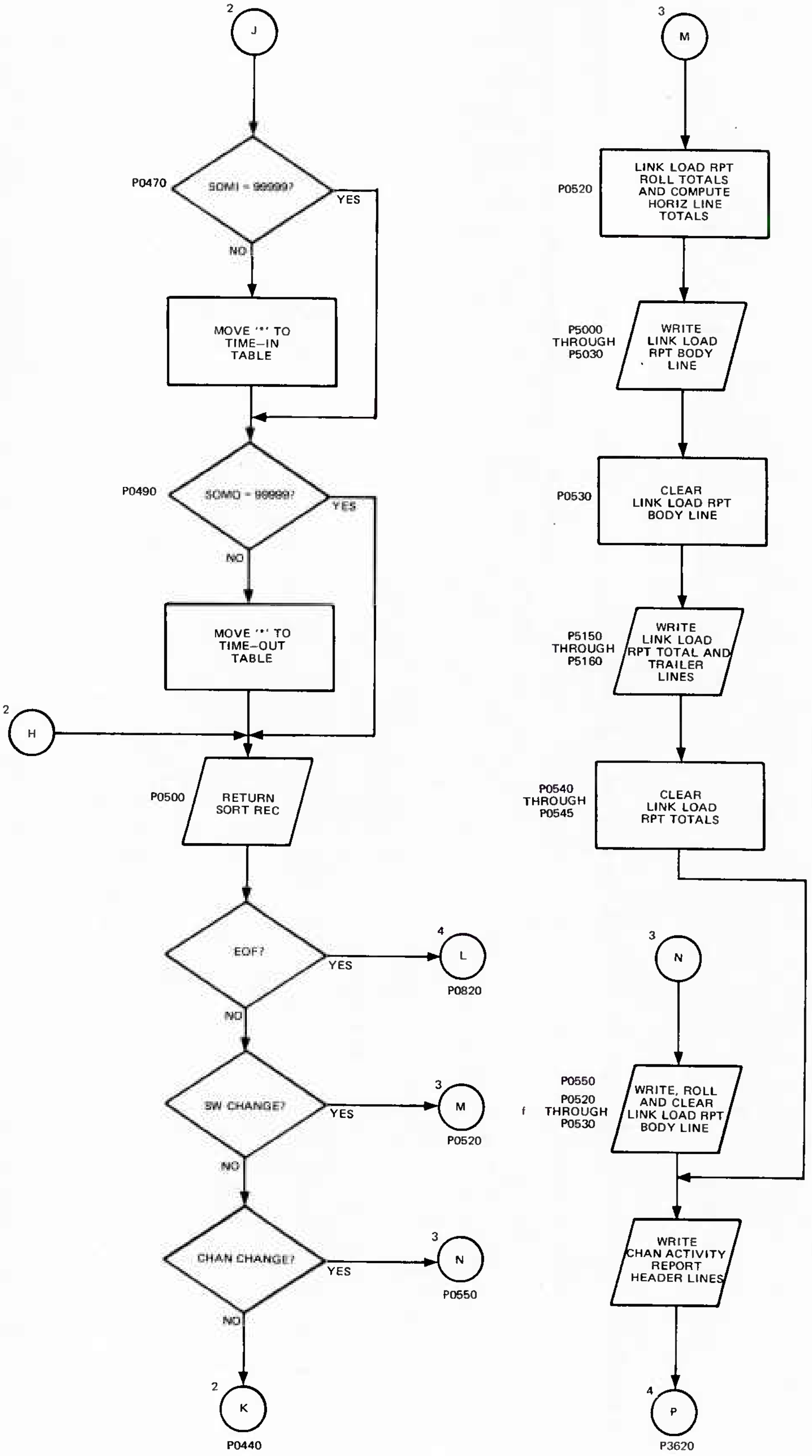
MNEMONIC REFERENCE	LEVEL	NUMBER OF CHARACTERS	ITEM DESCRIPTION
LINKFILE	FILE	80	Link File
LINKRECORD	RECORD	80	
FILLER	FIELD	9	Spaces
LINK-NAME	GROUP	6	Link Name
FILLER	FIELD	1	Value = "L"
L-ORIG	FIELD	2	Switch Name 1: C2 - C3
FILLER	FIELD	1	Value = "L"
L-DEST	FIELD	2	Switch Name 2: C2 - C3
FILLER	FIELD	1	Spaces
NR-CHANS	FIELD	3	Number of Channels
FILLER	FIELD	1	Spaces
SECUR	FIELD	1	Security Indicator
FILLER	FIELD	3	Spaces
TRANS-DLY	FIELD	5	Transmission Delay
FILLER	FIELD	1	Spaces
EOM-ACK-DLY	FIELD	5	End of Message Acknowledgement Delay
FILLER	FIELD	1	Spaces
REPT-SW	FIELD	3	Switch Name 1
FILLER	FIELD	4	Spaces
LTC-1	FIELD	1	LTC Reference 1
FILLER	FIELD	1	Spaces
DEST-SW	FIELD	3	Switch Name 2
FILLER	FIELD	4	Spaces
LTC-2	FIELD	1	LTC Reference 2
FILLER	FIELD	1	Spaces
QUE-DELY-1T02	FIELD	5	Average Starting Queue Switch 1 to 2
FILLER	FIELD	1	Spaces
QUE-DELY-2T01	FIELD	5	Average starting Queue (2 to 1)
FILLER	FIELD	15	Spaces


START
P0000 THROUGH P0014
INITIALIZE TABLE AND STORAGE AREAS
P0020 THROUGH P0090
READ SW-RADAY CARDS
VALIDATE AND LOAD SW-RADAY DATA INTO SW-RADAY-TAB
ERRORS?
YES
DISPLAY ERROR MSG
NO
P0100 THROUGH P0140
READ AMIE-LINK RECORDS
P0090
STOP
VALIDATE AND LOAD AMIE-LINK DATA INTO AMIE-TAB
STORE POINTER AND BLOCK SIZE OF EACH AMIE-TAB BLOCK IN SW-RADAY-TAB
ERRORS?
YES
DISPLAY ERROR MSG
1
A
P0200
1
A
SORT INPUT PROCEDURE f(P0170)
P0200
READ HEXLINK RECORD
EOF?
YES
2
D
P0260
NO
P0210 THROUGH P0225
SOMI-D = RADAY FOR SW?
P0230
YES
CONVERT SOMI AND SOMO TO MINUTES
1
C
P0250
NO
COMPARE SOMI-D : RADAY-1 SOMO-D : RADAY f(REPORTING SW)
BOTH = ?
NO
1
A
P0200
YES
P0240
CONVERT SOMO TO MIN MOVE 9999 TO SOMI
1
C
P0250
BUILD SORT RECORD
RELEASE SORT REC
1
A
P0200

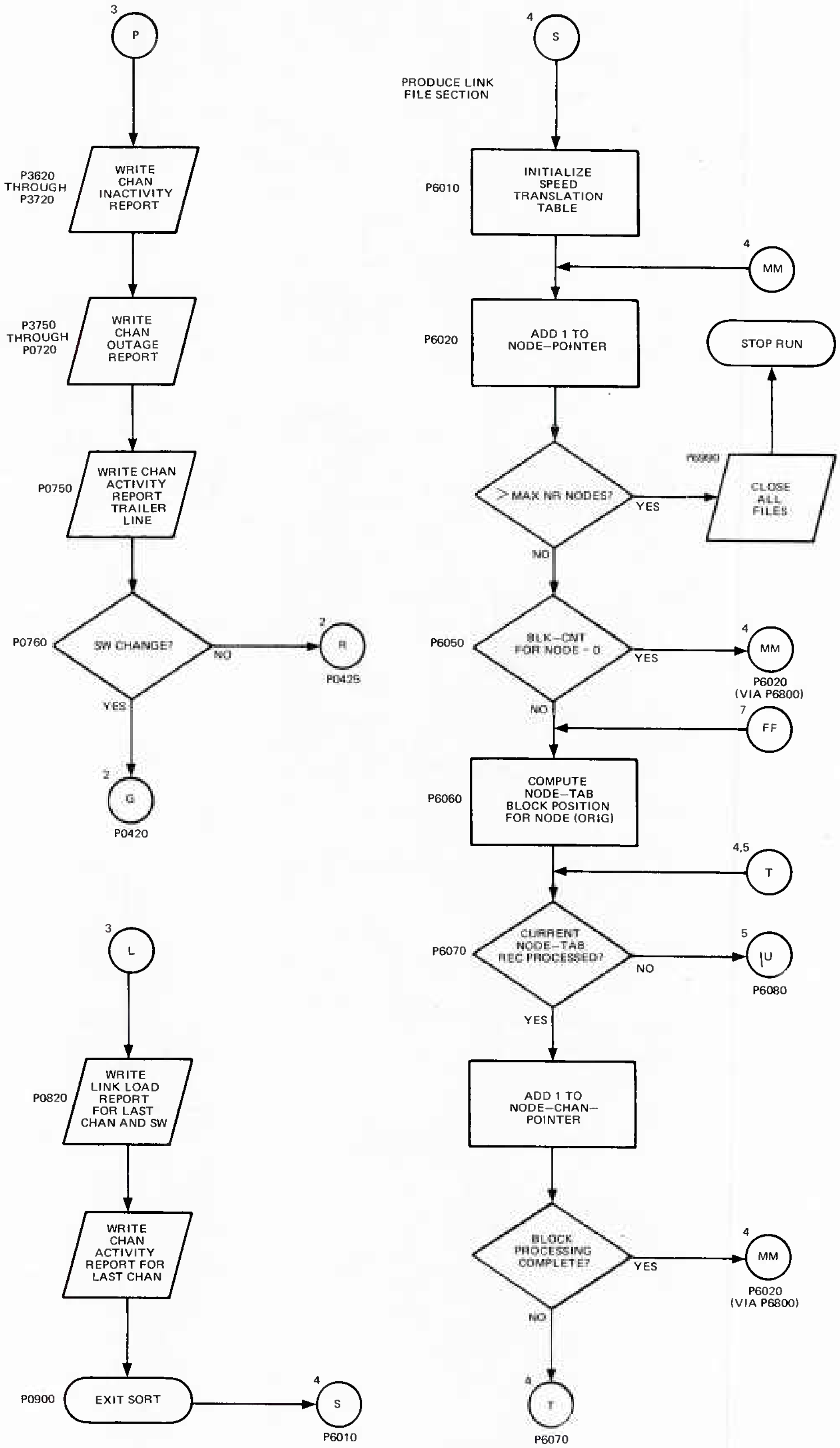
Link Load Program (Sheet 1 of 7)



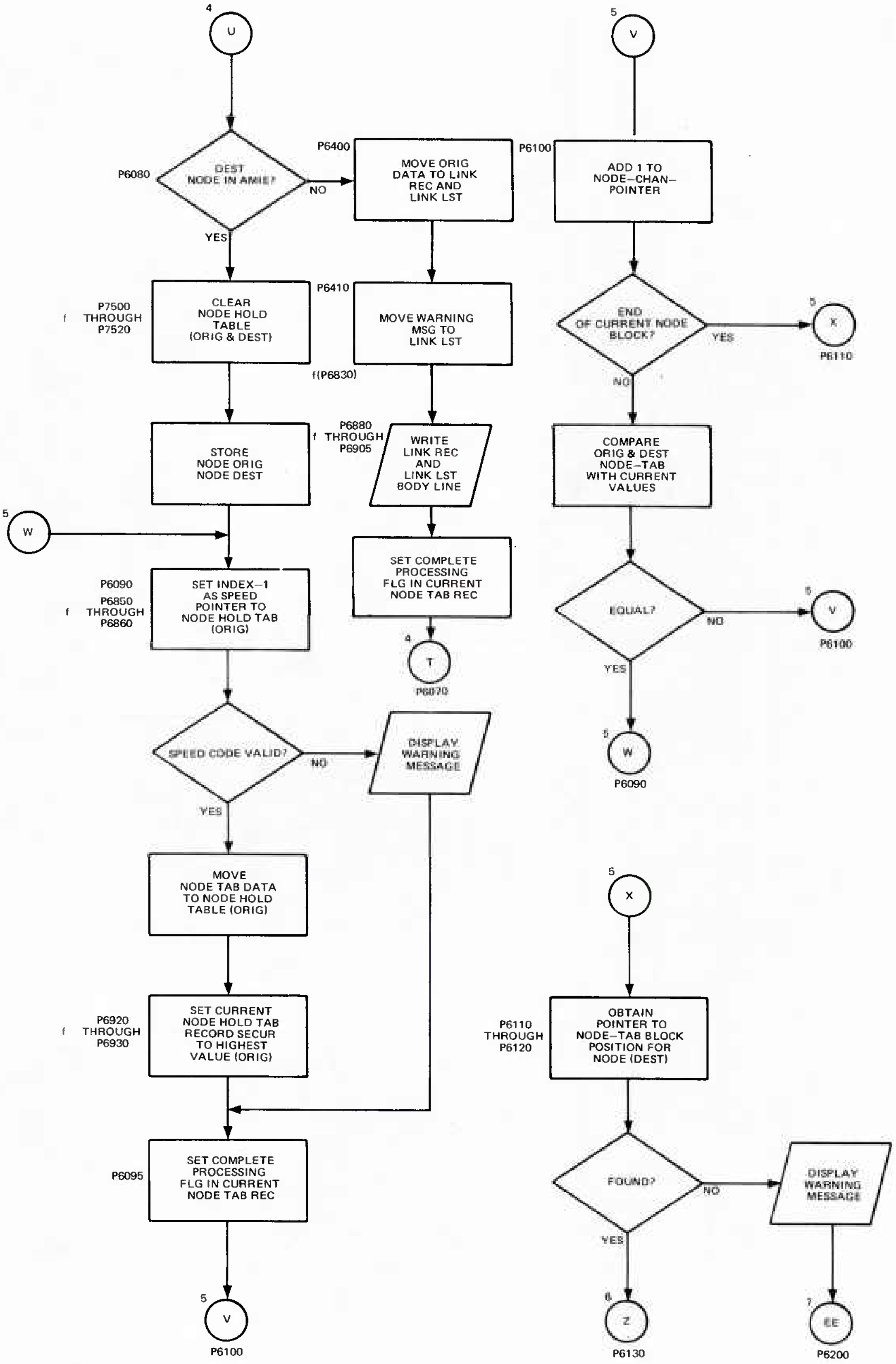
Link Load Program (Sheet 2 of 7)


2
J
P0470
SOMI = 99999?
YES
NO
MOVE '*' TO TIME–IN TABLE
P0490
SOMO = 99999?
YES
NO
MOVE '*' TO TIME–OUT TABLE
2
H
P0500
RETURN SORT REC
EOF?
YES
4
L
P0820
NO
SW CHANGE?
YES
3
M
P0520
NO
CHAN CHANGE?
YES
3
N
P0550
NO
2
K
P0440
3
M
P0520
LINK LOAD RPT ROLL TOTALS AND COMPUTE HORIZ LINE TOTALS
P5000 THROUGH P5030
WRITE LINK LOAD RPT BODY LINE
P0530
CLEAR LINK LOAD RPT BODY LINE
P5150 THROUGH P5160
WRITE LINK LOAD RPT TOTAL AND TRAILER LINES
P0540 THROUGH P0545
CLEAR LINK LOAD RPT TOTALS
3
N
P0550
P0520 THROUGH P0530
WRITE, ROLL AND CLEAR LINK LOAD RPT BODY LINE
WRITE CHAN ACTIVITY REPORT HEADER LINES
4
P
P3620

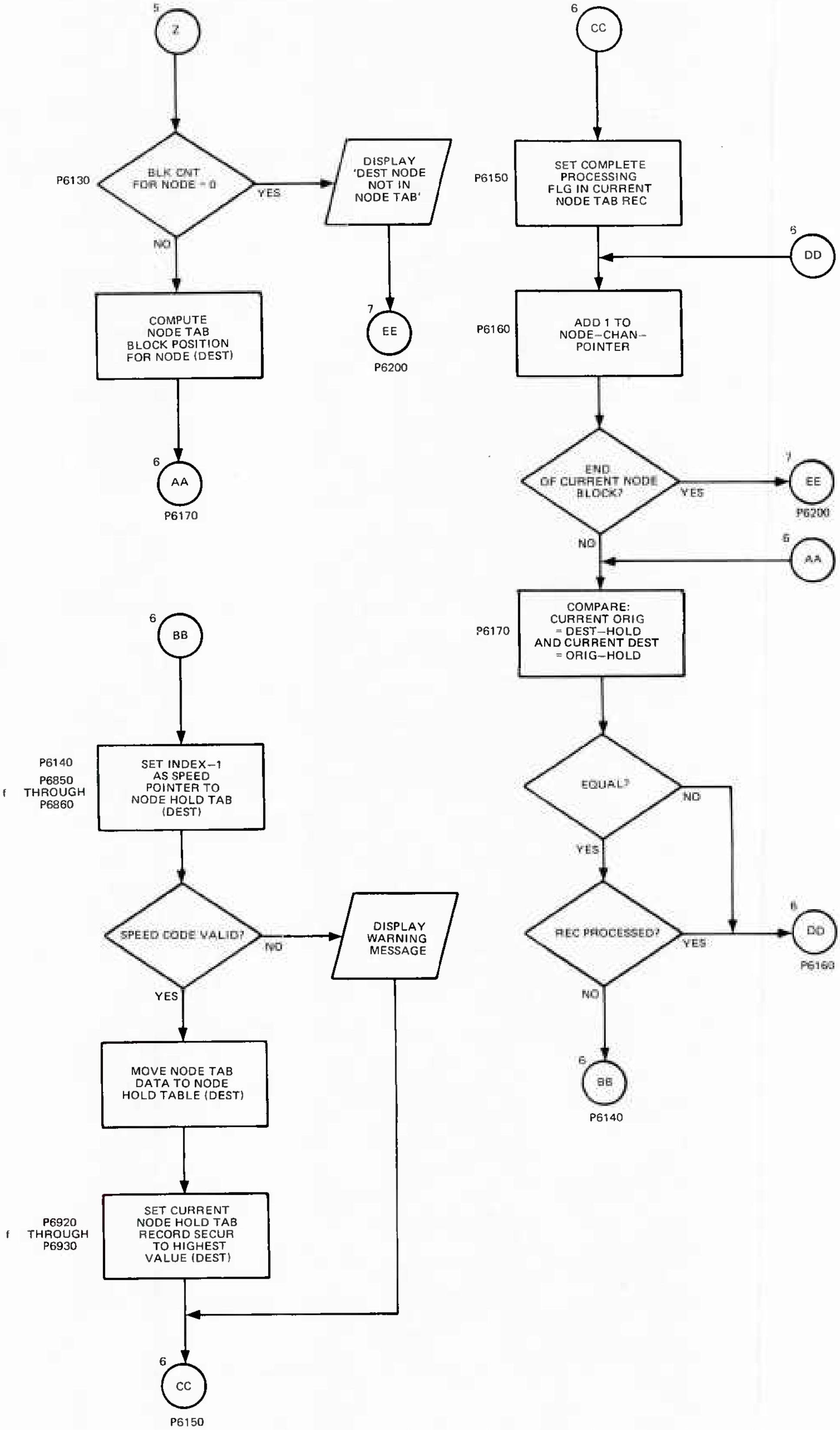
Link Load Program (Sheet 3 of 7)


3
P
P3620 THROUGH P3720
WRITE CHAN INACTIVITY REPORT
P3750 THROUGH P0720
WRITE CHAN OUTAGE REPORT
P0750
WRITE CHAN ACTIVITY REPORT TRAILER LINE
P0760
SW CHANGE?
NO
2
R
P0425
YES
2
G
P0420
3
L
P0820
WRITE LINK LOAD REPORT FOR LAST CHAN AND SW
WRITE CHAN ACTIVITY REPORT FOR LAST CHAN
P0900
EXIT SORT
4
S
P6010
4
S
PRODUCE LINK FILE SECTION
P6010
INITIALIZE SPEED TRANSLATION TABLE
4
MM
P6020
ADD 1 TO NODE–POINTER
STOP RUN
P6900
> MAX NR NODES?
YES
CLOSE ALL FILES
NO
P6050
BLK–CNT FOR NODE = 0
YES
4
MM
P6020 (VIA P6800)
NO
7
FF
P6060
COMPUTE NODE–TAB BLOCK POSITION FOR NODE (ORIG)
4,5
T
P6070
CURRENT NODE–TAB REC PROCESSED?
NO
5
U
P6080
YES
ADD 1 TO NODE–CHAN–POINTER
BLOCK PROCESSING COMPLETE?
YES
4
MM
P6020 (VIA P6800)
NO
4
T
P6070

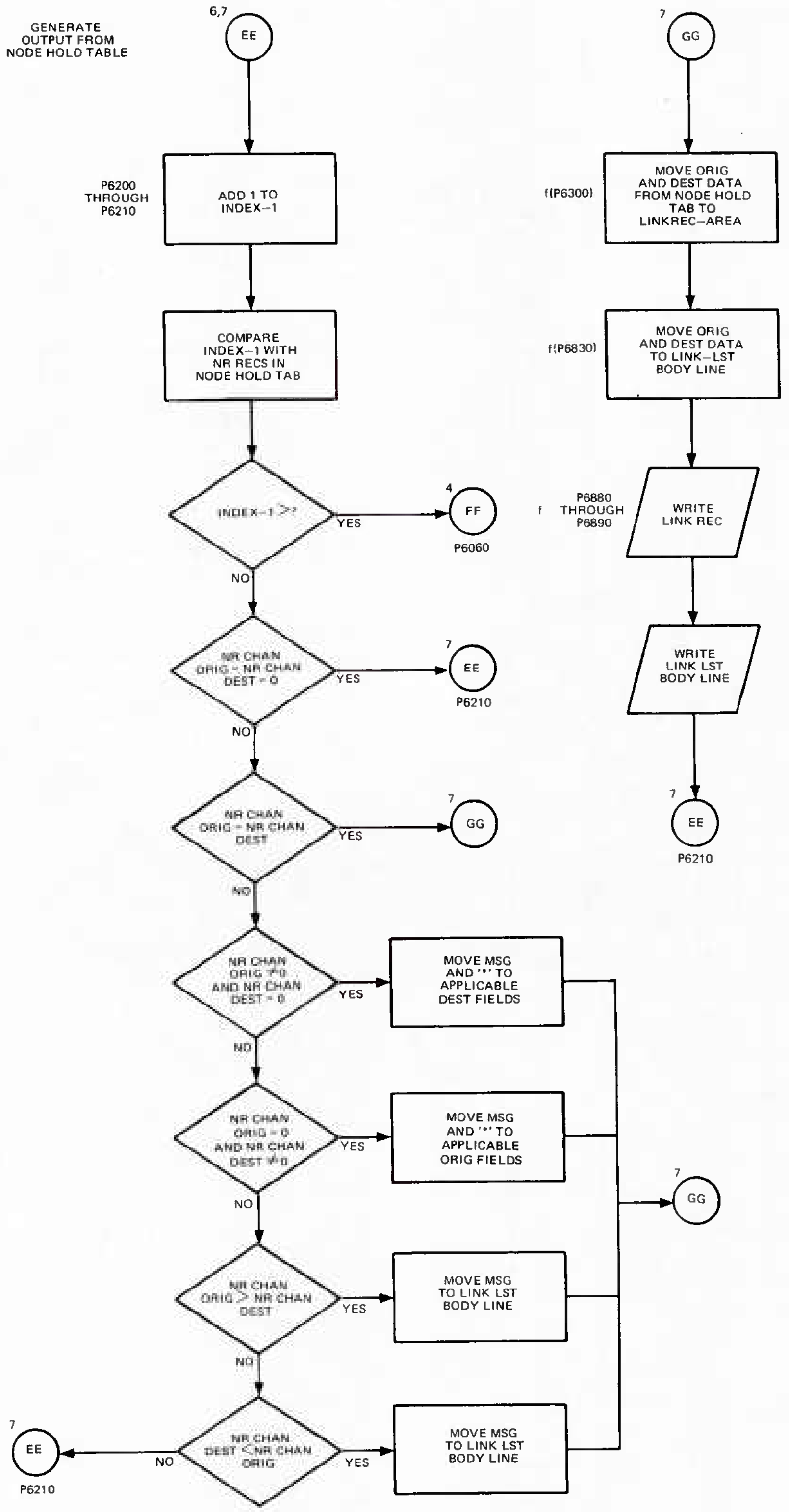
Link Load Program (Sheet 4 of 7)


4
U
P6080
DEST NODE IN AMIE?
NO
YES
P7500
f THROUGH
P7520
CLEAR NODE HOLD TABLE (ORIG & DEST)
STORE NODE ORIG NODE DEST
5
W
P6090
P6850
f THROUGH
P6860
SET INDEX–1 AS SPEED POINTER TO NODE HOLD TAB (ORIG)
SPEED CODE VALID?
NO
YES
MOVE NODE TAB DATA TO NODE HOLD TABLE (ORIG)
P6920
f THROUGH
P6930
SET CURRENT NODE HOLD TAB RECORD SECUR TO HIGHEST VALUE (ORIG)
P6095
SET COMPLETE PROCESSING FLG IN CURRENT NODE TAB REC
5
V
P6100
P6400
MOVE ORIG DATA TO LINK REC AND LINK LST
P6410
MOVE WARNING MSG TO LINK LST
f(P6830)
P6880
f THROUGH
P6905
WRITE LINK REC AND LINK LST BODY LINE
SET COMPLETE PROCESSING FLG IN CURRENT NODE TAB REC
4
T
P6070
DISPLAY WARNING MESSAGE
5
V
P6100
ADD 1 TO NODE–CHAN–POINTER
END OF CURRENT NODE BLOCK?
YES
NO
5
X
P6110
COMPARE ORIG & DEST NODE–TAB WITH CURRENT VALUES
EQUAL?
NO
5
V
P6100
YES
5
W
P6090
5
X
P6110
THROUGH
P6120
OBTAIN POINTER TO NODE–TAB BLOCK POSITION FOR NODE (DEST)
FOUND?
NO
YES
DISPLAY WARNING MESSAGE
6
Z
P6130
7
EE
P6200

Link Load Program (Sheet 5 of 7)


5
Z
P6130
BLK CNT FOR NODE = 0
YES
DISPLAY 'DEST NODE NOT IN NODE TAB'
7
EE
P6200
NO
COMPUTE NODE TAB BLOCK POSITION FOR NODE (DEST)
6
AA
P6170
6
BB
P6140
P6850 THROUGH P6860
SET INDEX-1 AS SPEED POINTER TO NODE HOLD TAB (DEST)
SPEED CODE VALID?
NO
DISPLAY WARNING MESSAGE
YES
MOVE NODE TAB DATA TO NODE HOLD TABLE (DEST)
P6920 THROUGH P6930
SET CURRENT NODE HOLD TAB RECORD SECUR TO HIGHEST VALUE (DEST)
6
CC
P6150
6
CC
P6150
SET COMPLETE PROCESSING FLG IN CURRENT NODE TAB REC
6
DD
P6160
ADD 1 TO NODE-CHAN-POINTER
END OF CURRENT NODE BLOCK?
YES
7
EE
P6200
NO
6
AA
P6170
COMPARE: CURRENT ORIG = DEST-HOLD AND CURRENT DEST = ORIG-HOLD
EQUAL?
NO
YES
REC PROCESSED?
YES
6
DD
P6160
NO
6
BB
P6140

Link Load Program (Sheet 6 of 7)


GENERATE OUTPUT FROM NODE HOLD TABLE
6,7
EE
P6200 THROUGH P6210
ADD 1 TO INDEX–1
COMPARE INDEX–1 WITH NR RECS IN NODE HOLD TAB
INDEX–1 >?
YES
4
FF
P6060
NO
NR CHAN ORIG = NR CHAN DEST = 0
YES
7
EE
P6210
NO
NR CHAN ORIG = NR CHAN DEST
YES
7
GG
NO
NR CHAN ORIG ≠0 AND NR CHAN DEST = 0
YES
MOVE MSG AND '*' TO APPLICABLE DEST FIELDS
NO
NR CHAN ORIG = 0 AND NR CHAN DEST ≠0
YES
MOVE MSG AND '*' TO APPLICABLE ORIG FIELDS
NO
NR CHAN ORIG > NR CHAN DEST
YES
MOVE MSG TO LINK LST BODY LINE
NO
NR CHAN DEST < NR CHAN ORIG
YES
MOVE MSG TO LINK LST BODY LINE
NO
7
EE
P6210
7
GG
f(P6300)
MOVE ORIG AND DEST DATA FROM NODE HOLD TAB TO LINKREC–AREA
f(P6830)
MOVE ORIG AND DEST DATA TO LINK–LST BODY LINE
P6880 THROUGH P6890
WRITE LINK REC
WRITE LINK LST BODY LINE
7
EE
P6210
7
GG

Link Load Program (Sheet 7 of 7)

SECTION 15 - FORTRAN SIMULATOR PREPROCESSOR PROGRAM

15.1 PROGRAM REQUIREMENTS DESCRIPTION

15.1.1 Program Name: FORTRAN Simulator Preprocessor

15.1.2 Program Identification: CSCSCRPT

15.1.3 Purpose

This program facilitates the writing of complex FORTRAN programs which use bit manipulation and multiple references to structured tables. Bit manipulation is not a feature of standardized FORTRAN IV; however, most compilers make some provision for accomplishing it. To make program source coding reasonably machine independent, some means other than direct coding of bit manipulation is required. Also, program maintenance can be extremely difficult when structured tables are used. For example, one minor change to a table could require thousands of statements to be reviewed for possible reference to the item to be modified. The Preprocessor program permits a source program deck to contain pseudo-instructions that address structured tables and items requiring bit manipulation using mnemonic references. The program source coding deck is passed through the Preprocessor program where the pseudo-instructions are replaced by instructions applicable to the computer on which the program is to be operated. See Appendix C for a more complete discussion of this program.

15.1.4 Requirements

The program is written in FORTRAN IV language and requires approximately 165K bytes of core when executed on an IBM 360/65.

15.2 INPUT SPECIFICATIONS

There are normally two inputs to this program. One is a FORTRAN source deck containing pseudo-instructions and mnemonic references. The other is a set of tables which define the mnemonic references. See Appendix C for a more complete discussion of these inputs.

15.3 OUTPUT SPECIFICATIONS

The output file from the program is a FORTRAN source deck in which the pseudo-instructions in the input file have been replaced by FORTRAN instructions valid for some particular computer system. The program also generates a listing of the input files.

15.4 PROGRAM LOGIC

The program is capable of performing three different functions; inserting nonexecutable statements in a source deck, executing a preset list of valid FORTRAN instructions (comparable to a macro function), and manipulating data which has been addressed by a mnemonic reference.

15.4.1 Table Loading

Prior to program execution, a card file containing from one to three input tables must be constructed depending on the use of the pseudo-instructions. When the INSERT function is used, a set of punched cards identified with INSERT must be assembled which contain those nonexecutable instructions to be inserted. These cards may contain statement numbers in Columns 1 to 5. When the EXECUTE function is used, a set of punched cards identified with MACROS must be assembled which contain those executable instructions to be added to the source deck. These instructions may not have statement numbers; however, any statement number appearing on the psuedo-instruction execute card is transferred to the first card added to the source deck. The instructions included in the MACROS deck can include pseudo-instructions except for an INSERT or another EXECUTE. The third table is composed of a set of cards which define the mnemonic references to be used in the remaining psuedo-instructions. Included in the card is the actual address of the item and the reference word. Optional entries are a numeric value which represents the largest value this item may attain without error, and, when bit packing is used, the necessary items required to unpack the item from the remainder of the word. This punched card set is identified simply as TABLES.

When the program is executed these three tables are loaded into corresponding arrays.

15.4.2 Source Deck Conversion

The source deck is read a card at a time and a test made for the letter "P" in Column 1 which indicates a pseudo-instruction. If there is no "P," the card is transferred directly to the output file. If the card is a psuedo-instruction, Column 1 is changed to a "C" and the card is transferred to the output file. Next the card is tested for an INSERT or EXECUTE function. If found, the appropriate array is searched and any entries with the same reference are transferred to the output file. If none are found, an appropriate error message is inserted.

The exact processes involved in handling the remaining psuedo-instructions (INCREMENT, DECREMENT, SET, CLEAR, ADD, SUBTRACT, MOVE, and TEST) are somewhat machine dependent and therefore are not covered here. Consult the flow charts in Appendix C.

15.5 MACHINE DEPENDENCE

The preprocessor program, although written in FORTRAN IV, is completely machine dependent in the sense that it must be written to produce coding for a specific computer system.

15.6 FLOW CHARTS

Flow charts for the Preprocessor program will be found in Appendix C.

SECTION 16 - DATASERV PROGRAM

16.1 PROGRAM REQUIREMENTS DESCRIPTION

16.1.1 Program Name: DATASERV

16.1.2 Program Identification: DATASERV

16.1.3 Purpose

The DATASERV program provides the ability to manipulate card image data files. The program has four modes of operation: creating a new file, listing an existing file, modifying an existing file, and duplicating an existing file.

16.1.4 Requirements

The program is written in general FORTRAN IV language and requires approximately 60K bytes of core for execution on an IBM 360/65.

16.2 INPUT SPECIFICATIONS

The program may have one input file composed of 80-byte punched card image records. The program requires an input control card file to determine mode of operation. Table 16-1 contains the control card format. Depending on the mode of operation, this input may be a mixture of control and data cards.

16.3 OUTPUT SPECIFICATIONS

Depending on the mode of operation, the program may produce an output file composed of 80-byte punched card image records. The program may or may not produce a listing of an involved data set, depending upon the options selected in the Control Card Deck. The program does produce a listing of the control cards and, in the Modify Mode, a listing of all deleted cards.

16.4 PROGRAM LOGIC

The mode of operation is determined by the first card in the control deck.

16.4.1 Creating a New File

When the first card specifies NEW FILE, the program assumes that a new output file is to be created from the data cards following the control card. All cards are written on the new file until an END control card or the end of the input deck is reached. A listing of the cards in the new file is produced. The listing includes the sequential number of each card in the file.

16.4.2 Listing an Existing File

When the first control card specifies LIST, the cards in the input file are listed, including the sequential number of each card.

16.4.3 Modifying an Existing File

When the first control card specifies DELETE, ADD, DELASS, or NO LIST, the program assumes the Modify Mode. In this mode an existing file is read and a new output file is generated which has been modified by deleting specified cards, adding cards in the places specified, or deleting specified cards and adding others in their place. This mode normally produces an output listing of the new file including the sequential number of each card. This listing may be inhibited by the inclusion of a NO LIST control card.

16.4.4 Duplicating an Existing File

An existing file may be duplicated under another name and/or on another device by specifying MOVE in the first control card. If a listing of this file is desired, the control card may save MOVE LIST. Program logic flow charts are included at the end of this section.

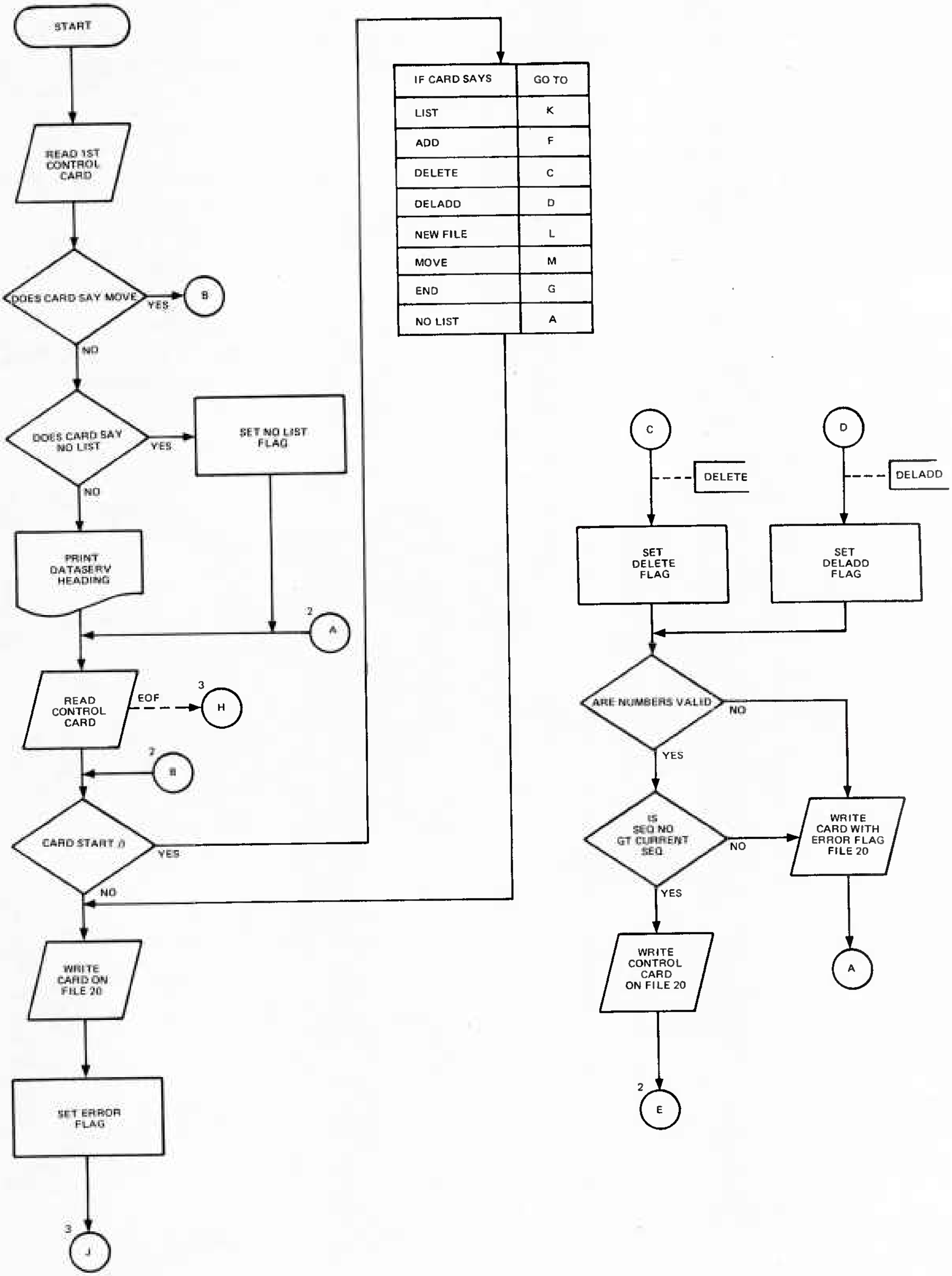
Program logic flow charts are included at the end of this section.

16.5 MACHINE DEPENDENCE

The program is written in general FORTRAN IV language and is essentially machine independent.

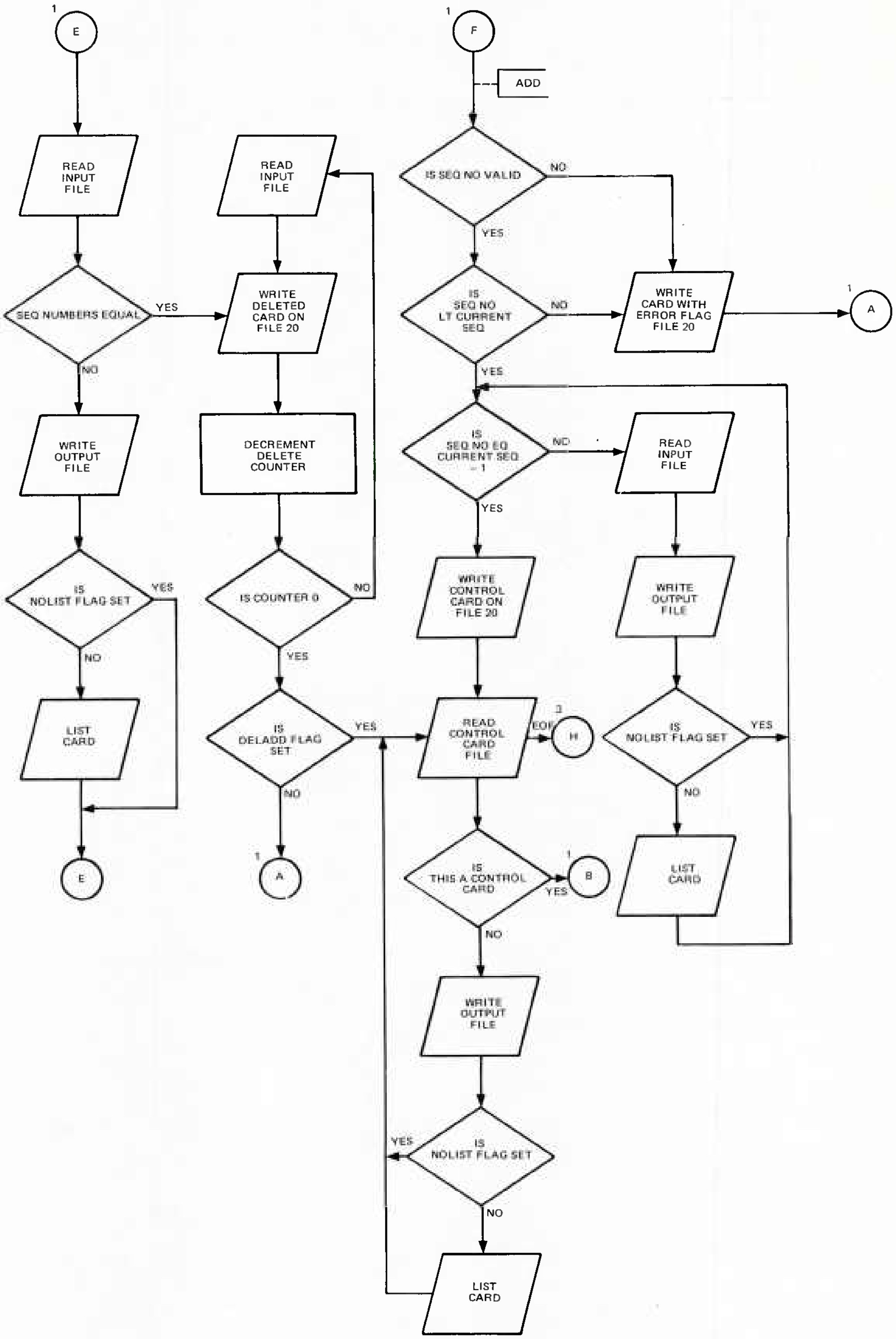
Table 16-1. DATASERV Program Control Card Format

CARD COLUMN	DESCRIPTION	EXPLANATION
1-2	Control Card Identifier	Contains the characters "/)" punched in EBCDIC
3	Blank	
4-80	Control Statements	Acceptable control statements are NEW FILE, LIST, MOVE, MOVE LIST, DELETE, ADD, DELADD, NO LIST and END. These words start in cc 4. DELETE and DELADD should be followed by two numerical values separated by a comma. These values may be anywhere between cc 11 and 80. ADD should be followed by one numerical value. This value may be anywhere between cc 8 and 80.

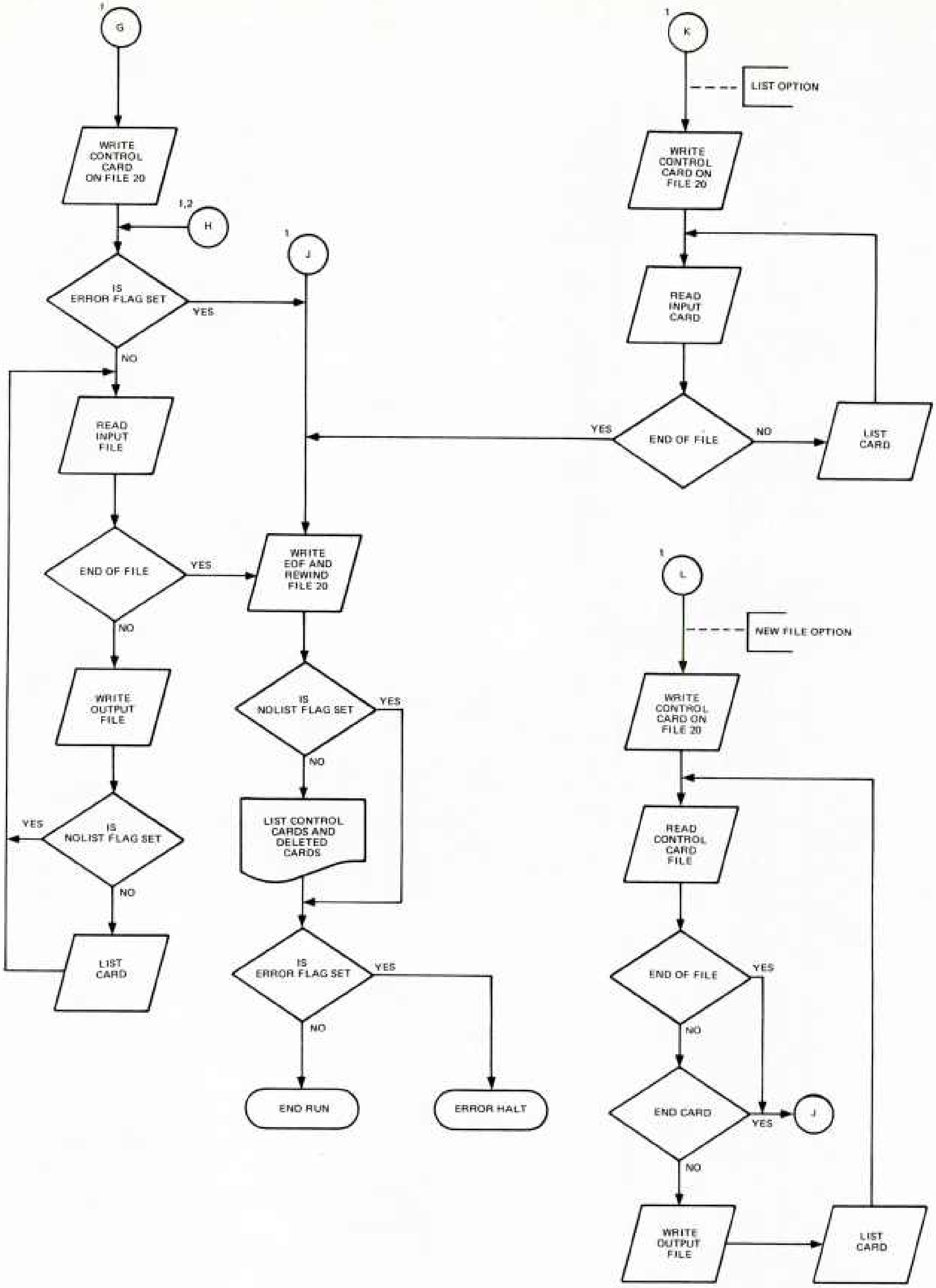


IF CARD SAYS	GO TO
LIST	K
ADD	F
DELETE	C
DELADD	D
NEW FILE	L
MOVE	M
END	G
NO LIST	A

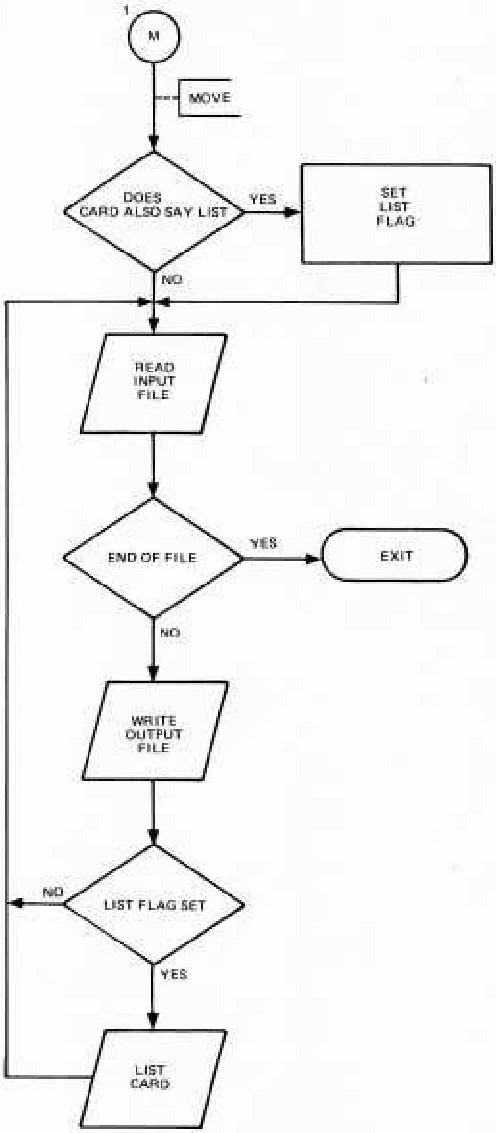
DATASERV Program (Sheet 1 of 4)



DATASERV Program (Sheet 2 of 4)



DATASERV Program (Sheet 3 of 4)



DATASERV Program (Sheet 4 of 4)

REFERENCES

1. DIN Simulator User's Manual, CSC Report No. 413681-1-1 of May 72

2. Overseas AUTODIN Simulator, Final Report, C&S Report No. R403701-1-3, December 1967.

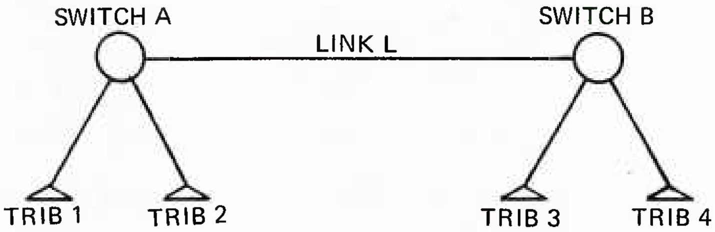
3. Overseas AUTODIN Simulator, Program Report, C&S Report No. R403701-1-4, December 1967.

4. Automated AUTODIN Traffic Processing System, Program Report, C&S Report No. R408701-69-1, May 1969.

5. Automated AUTODIN Traffic Processing System, User's Manual, CSC Report No. R407595-1-1, July 1970.

APPENDIX A - BASIC SIMULATION SYNOPSIS

The operation of the Basic Simulation program can best be explained by introducing a message into the network being simulated and following the message to its destination. Consider the network to be the simple one illustrated below:



When the program is initialized, the Event Message file is searched for the first input which occurs within the period of simulation and enters an event in the calendar of events reflecting the time the event is to occur. A message input contains the originating tributary, destination tributary(ies) and message characteristics (length, type, priority, and security).

When the message event reaches the head of the calendar of events list, the program branches to the Message Input Section. A Message Status Table, containing message number, length, type, precedence, security, and destinations, is generated for the incoming message. If the message has more than two destinations, one or more Message Continuation Status Tables are also generated. For this explanation assume that the message was originated by TRIB1, is destined for TRIB2, TRIB3, and TRIB4, and is an unclassified teletype message of routine precedence with a length of 25 line blocks. The program scans the input Channel Status Table(s) of TRIB1 for an available input channel on which the message can be transmitted to SWITCHA. If no channel is available, the message is added to the Input Channel Queue of the Tributary Status Table (TRINQ) by precedence, and the Tributary Input Queue Counters are incremented in the Tributary Status Table (TRIQ) and the Node Status Table (NSQIT). Assuming a channel to be available, an End of Message Trib to Node Event (Type 5) is generated and placed in the Calendar of Events. The duration of the event is computed from the length of the message (MSLN of Message Status Table) and the speed of the transmission line (for this message, TRSPT of Tributary Status Table). The channel is marked as busy by setting the Start Of Message indicator in the Channel Status Table (CHSM). The Transmission Event Address is also referenced in the same table (CHEV). The next event on the Event Message file is read in and an appropriate event entered in the Calendar of Events.

When the transmission delay has expired, the Event Type 5 section is executed. The event is placed in the Message Processor Input Action List of SWITCHA's Node Status Table (MPICS-MPICE) and the End Of Message indicator is set in the Channel Status Table (CHEM). If an acknowledgement delay is specified for TRIBA, an Acknowledgement Delay Event (Type 8) is generated and placed in the calendar. If no delay is specified, the Channel Delay Flag is set in the Channel Status Table (CHDFLG).

The next action taken depends upon whether the Message Processor cycle for SWITCHA passes through its Input function first or whether the Acknowledgement Delay expires first. This would be determined by the parameters specified for the switch and tributary. Assume for this explanation that the Input function for SWITCHA occurs first. In this portion of the program, the Message Processor Input Action List (MPICS) is sampled to determine if there are messages waiting to enter the switch. Having found a message, a Routing Event (Type 9) is generated and placed in the Calendar of Events. The time of this event depends on the precedence of the message. Two routing parameters are entered in the Node Status Table for each switch; the time to route a high precedence message (MPIRIH) and the time to route a low precedence message (MPIRIL). Next, the Channel Delay Flag of the Channel Status Table (CHDFLG) is tested. If the flag is not set, as would be the case in this explanation, the flag is set and the program exits from the Input cycle. If the flag had been set, as would be the case if no acknowledgement delay were specified, or if the delay had expired, the channel would have been cleared and the Tributary Input Queue (TRINQ) would be tested for waiting messages.

When the End of Acknowledgement Delay Event (Type 8) is executed, the same action occurs. If the Channel Delay Flag is not set, it is set and an exit made from the routine. If the flag is set, the channel is cleared and made available for the next transmission.

At the appropriate time the Routing Event (Type 9) is executed. In the Routing routine the destinations in the Message Status Table (MSDN1-MSDN2), and the Message Continuation Table (MCDN1-MCDN6), when necessary, are examined to determine how message delivery can be effected. In this example, it would be determined that TRIB2 was located at the current switch and that the message would have to be transmitted on LINK1 to be delivered to TRIB3 and TRIB4. A duplicate Message Status Table is generated and placed in the output queue of TRIB2's Tributary Status Table (LTQS-LTQE) and another Message Status Table with destinations of TRIB3 and TRIB4 is placed in the output queue of LINK1's Link Status Tables (LTQS-LTQE). The appropriate queue counters are incremented in the Node Status Table (NSQL and NSQOT), TRIB2's Tributary Status Table (TROQ), and in LINK1's Link Status Table (LNQLO). Queue Linkage Tables are generated which point to TRIB2 and LINK1 and are placed in the Message Processor Output Action List of the Node Status Table (MPOQS-MPOQE).

When the MP Cycle Event (Type 2) passes through the next Output function, the pointers in the Message Processor Output Action List causes the Output Channel Status Tables for TRIB2 and LINK1 to be searched for available channels. Assuming that there are free channels, a Tributary Transmission Event (Type 4) is generated for the TRIB2 channel and Link Transmission Event (Type 3) for the LINK1 channel. The messages are not removed from their respective queues at this time but are marked as being transmitted by setting the Message Transmission Indicator in the Message Status Table (MSXF). Message transmission does not actually begin at this time since the output processing delay must be accounted for. To effect this delay the events are placed in the Channel Release List of the Node Status Table (MPORL).

The next function of the MP Cycle Event (Type 2) for SWITCHA is the Release function which occurs after the output processing delay. At this time the events are removed from the Channel Release List (MPORL), the transmission delays computed, and the events entered in the calendar.

When the Tributary Transmission Event (Type 4) is executed, indicating that delivery to TRIB2 has been completed, the associated Message Status Table is removed from the Tributary Status Table Output Queue (LTQS-LTQE) and its memory locations are returned to dynamic storage. If no acknowledgement delay has been specified for TRIB2, the channel is cleared and available for the next transmission. If a delay is specified, a Delay Event (Type 7) is generated and the channel is cleared when executed.

When the Link Transmission Event (Type 3) is executed, the ensuing actions at SWITCHB are essentially the same as when the message arrived at SWITCHA from TRIB1, therefore, those descriptions are not repeated.

The foregoing process may be slightly varied if the LTC/ADU function is being simulated or if a limit has been placed on the number of messages permitted in outgoing queues. In the LTC/ADU function, when the message enters the system not only must there be an available channel but also sufficient space in the LTC/ADU to which the tributary is connected. This is done by determining the LTC number from the Tributary Status Table (LTLTC) and testing that LTC's Throttle Indicator in the Node Status Table (FETI1, FETI2, FETI3 or FETI4). If the indicator is not set, transmission may begin. The Line Block Count of the message (length) is subtracted from the Line Blocks Available Count for that LTC (FELBA1, FELBA2, FELBA3, or FELBA4) and a test is made to see if the reduced value now falls below the throttle threshold (FETT). If it does, the appropriate throttle indicator is set.

When the End of Transmission Event (Type 2) is executed, the Line Blocks are returned to the Line Blocks Available Count. If the LTC is in the throttled state, a test is made against the Unthrottle Threshold (FETU) to determine if the LTC can resume accepting messages (the throttle indicator is cleared).

Similar actions take place on message transmission from the switch to the destination tributary. In link transmissions, both source and destination LTCs must be tested.

If a message cannot enter the system due to a throttled LTC, a Queue Linkage Table is generated and placed in the MP Tributary Input List in the Node Status Table (MPIQS-MPIQE). During the MP Input cycle a test is made to see if the LTC has become unthrottled and transmission can begin.

Whenever a limit has been placed on the number of messages in queue at a switch, each time a message is placed in queue the Fixed Queue Counter in the Node Status Table is decremented (MPOFQ). When an outgoing message transmission is completed, the counter is incremented. Thus, before a message is routed, this counter must be tested to determine if sufficient queue space is available. This is accomplished during the MP Input cycle. Before a Routing Event is generated, the counter is tested and if sufficient space is not available, the message is placed in the MP Tape Queue List of the Node Status Table (MPTQS-MPTQE) and the Tape Queue Counter is incremented (MPTQC). Thereafter, the first step in the MP Input cycle is to test the messages in the tape queues to see if they can now be routed.

If the simulator is being operated in the EDIT mode, the chain of events is somewhat abbreviated since the switch cycle function is not simulated. The actions are the same for the introduction of message into the system. The transmission delay is generally longer because an Effective Transfer Rate is applied which takes into account transmission time, acknowledgement delay, and switch cycle functions.

When the End of Transmission Trib to Node Event (Type 5) is executed, a Routing Event (Type 9) is generated and placed in the Calendar Of Events. Since there can be no acknowledgement delay, the channel is released immediately for the next incoming messages.

During the execution of the Routing Event procedure, as soon as a message is placed in an outgoing queue, the routine branches to an abbreviated version of Process Output Data function and onward transmission begins immediately if there is no queue.

APPENDIX B - BASIC SIMULATION EVENTS

The following is a list of the events and a synoposis of their actions:

Event Type 1 - This is a cancelled event and serves only to return the memory locations utilized by the event dynamic storage. It is used when an event, already in the Calendar Of Events, must be cancelled due to some other event, such as an outage. Rather than search the calendar for the event and remove it, its type is changed to a one and it is cancelled when it reaches the head of the chronological queue.

Event Type 2 - This event is used to control the cycle at each switch whenever the simulator is being operated in a cyclic mode. When executed, it causes one of the four switch functions to be executed (Process Input Data, Process Output Data, Release Output Data, or Process Miscellaneous Data). When one function has been completed, the event is advanced to the next function and is returned to the calendar with the scheduled time of the next function. There is one event of this type for each switch.

Event Type 3 - This event signals the end of transmission of a message on an interswitch link. When operating in a cyclic mode, the message is made available to the destination switch and is further processed when the next Process Input Data function of that switch is executed. In addition, if an acknowledgement delay has been specified, an Acknowledgement Delay Event (Type 6) is generated. When operating in a noncyclic mode, this event generates a Routing Event (Type 9) for the message just completed and begins transmission of the next message in queue (if any).

Event Type 4 - This event occurs at the end of transmission of a message from a switch to a destination tributary. A message delivery descriptor is written on an output file. When operating in the cyclic mode and an acknowledgement delay is specified, an Acknowledgement Delay Event (Type 7) is generated. If no delay is specified, the tributary output channel becomes available for the next transmission when the switch executes its next Process Output Data function. When operating in a noncyclic mode, immediate transmission is initiated for the next message in queue (if any).

Event Type 5 - This event occurs at the end of transmission of a message from a tributary to a switch. When operating in a cyclic mode, the message is made available to the switch and is further processed when the next Process Input Data function of that switch is executed. If an acknowledgement delay has been specified, an Acknowledgement Delay Event (Type 8) is generated. When operating in a noncyclic mode, this event generates a Routing Event (Type 9) for the message and begins transmission of the next message in queue (if any).

Event Type 6 - This event occurs at the end of an Acknowledgement Delay on an interswitch link channel. Providing the message carried on the previous transmission has been accepted by the destination switch, the channel is made available to the source switch and the next transmission is initiated on that switch's next Process Output Data function. If the message has not yet been accepted, the channel becomes free at the time the message is accepted.

Event Type 7 - This is the Acknowledgement Delay Event for a switch to destination tributary. Upon execution, the tributary channel is made available and another transmission may be initiated on the switch's next Process Output Data Function.

Event Type 8 - This is the Acknowledgement Delay Event for a tributary-to-switch transmission. Upon execution, the tributary channel is made available and transmission may begin for the next message in queue (if any).

Event Type 9 - This is the Routing Event. Upon execution, a message which has been accepted on a previous Process Input Data function is processed through the Routing Section and copies of the message are placed in the appropriate outgoing queues according to the specified Routing Doctrine.

Event Type 10 - When this event occurs the simulation process terminates.

Event Type 11 - Event Types 11 through 19 are triggered by inputs from the Event Message Input file. Event Type 11 signifies a message input and causes a message to be assembled from data on the input record. If there is a free channel from the tributary specified to its switch, message transmission begins. Otherwise the message is queued at the tributary according to priority and transmission begins when a channel becomes available.

Event Type 12 - This special event causes either a specified channel or an entire interswitch link or tributary to be placed out of service.

Event Type 13 - This special event restores either a specified channel or an entire interswitch link or tributary to service.

Event Type 14 - This special event causes an entire switch, including all its links and tributaries, to be placed out of service.

Event Type 15 - This special event restores to service a switch, including its tributaries. The links are also restored if their destination switches are not out of service.

Event Type 16 - This special event permits the changing of Routing Doctrine during a simulation.

Event Type 17 - This special event changes the Altroute Indicator for a specific link from OFF to ON or from ON to OFF. When this indicator is ON it permits altrouting of low priority messages during an outage.

Event Type 18 - This special event changes the preemption indicator for a specified link or tributary from OFF to ON or from ON to OFF. When this indicator is ON it permits Operational Immediate messages to preempt messages of lower priority.

Event Type 19 - This special event adds entries to or deletes entries from the Contingency Alternate Routing Plan (CARP) Tables.

Event Type 20 - When a queue delay parameter is specified for a link or tributary, low priority messages are not transmitted for the period of time specified. This event releases the low priority messages.

Event Type 21 - This event causes the Status Records for each switch to be written on an output file.

Event Type 22 - This event causes Status Records for each link to be written on an output file.

Event Type 23 - This event causes Status Records for each tributary with messages in queue to be written on an output file.

Event Type 24 - This event causes all core elements essential to restarting the simulator to be written on an output file.

APPENDIX C - FORTRAN SIMULATOR PREPROCESSOR PROGRAM FOR IBM 360

C.1 GENERAL

In any large program which uses multiple references to tables, it is preferable to use some form of table definition system which allows the tables to be modified without rewriting the entire program. It is also preferable, in terms of programming time, to use macro or pseudo-instructions to manipulate the data in the tables.

For ease in reprogramming and modifying the programs of the VON and DIN Simulators, a Preprocessor program has been developed which includes pseudo-instructions and table definitions. The program has reference lists of table items defined by mnemonics, their relative positions in the tables, and their bit structure in the relative word. When a pseudo-instruction is recognized by the Preprocessor, the necessary coding is added to extract information from or store information into the table item. Thus, if a table modification is required, the item(s) need only be redefined in the Preprocessor and the subject program processed and compiled.

The Preprocessor concept has another advantage. The Preprocessor is written for a specific computer whereas the main programs are written in basic FORTRAN language. Thus, to switch to another computer with a different word length and its own set of special instructions, only the Preprocessor would need modification.

C.2 PREPROCESSOR INPUTS

The Preprocessor performs three different functions, each of which requires an input to define the desired functions. The functions and their input files are:

1. INSERT Function

   The INSERT function allows nonexecutable statements to be inserted in the programs and subroutines without writing the entire set of cards in each. The input to the Preprocessor is a set of cards in which the first card says INSERT in Columns 1-6 and the last card says END in Columns 1-3. The remaining cards in the file contain the desired coding to be inserted in

Columns 1-72. Columns 73-78 contain the mnemonic reference by which the programmer addresses these instructions. When an INSERT pseudo-instruction is encountered in the source program, this input file is searched for a corresponding mnemonic reference and any matching card or cards are inserted into the program at that point.

2. MACRO Function

The MACRO function allows a series of short, frequently used executable statements to be inserted in the program without writing the full set each time. The input to the Preprocessor is a set of cards in which the first card says MACROS in Columns 1-6 and the last card says END in Columns 1-3. The remaining cards contain the desired coding to be inserted in Column 1 and Columns 6-72. Columns 73-78 contain the mnemonic reference by which the programmer addresses these instructions. When an EXECUTE pseudo-instruction is encountered in the source program, this input file is searched for a corresponding mnemonic and any matching card or cards are inserted into the program at that point. The cards in this file may not contain statement numbers in Columns 2-5. However, a statement number used in the EXECUTE pseudo-instruction is transferred to the first statement in the MACRO set. The MACRO instruction cards may contain other pseudo-instructions except another EXECUTE or an INSERT.

3. TABLE REFERENCE Function

This function allows reference to items in tables and arrays by mnemonic references in place of fixed coding in the programs. The input to the Preprocessor is a set of cards in which the first card says TABLES in Columns 1-6 and the last card says END in Columns 1-3. The format for the remaining cards for the IBM 360 is defined in Table C-1.

C.3 CARD FORMAT

Pseudo-instruction cards for the Simulator Preprocessor must be written in accordance with the format defined in Table C-2.

C.4 PSEUDO-INSTRUCTION TYPES

The Simulator Preprocessor currently recognizes ten different instructions. The pseudo-instruction generally contains a command and one or two references. The references may be a table reference, a variable, or, depending upon the instruction, an integer constant. The Preprocessor compares the reference(s) used in the instruction against the TABLE entries and if a match is found, the generated coding reflects the steps necessary to extract and/or use that table reference. If no match is found, the reference is considered to be a variable or integer constant. One of the fields in the TABLES file may be a maximum value which the item may not exceed. When such a maximum is used, some of the pseudo-instructions cause a test to be made of the new value of the item to assure that the value does not exceed the specified maximum. The test is such that should overflow occur, or if underflow occurs on a SUBTRACT or DECREMENT instruction, a PRINT statement is executed that shows what action caused the overflow or underflow. The overflow or underflow also causes an exit to Statement 9999. The programmer should insert his own coding with this statement number for program wrapup. The Preprocessor may generate, just prior to the END card in the source deck, statements which are numbered 9990 to 9994; therefore, these statement numbers must be reserved. Also, the generated coding may use the integer variables, ITMP1 and ITMP2. The programmer should avoid using these variables as the contents may be altered by a pseudo-instruction. The IBM 360 version of the Preprocessor may also use the variables MSK1 to MSK999, NM1 to NM999, and NMA1 to NMA999. When used, DATA statements are generated and inserted into the program. The program or subroutine END statement must begin in Column 7.

The pseudo-instructions are:

	Column 7	Column 16
1.	EXECUTE	Reference

This instruction operates as described in Paragraph C.2.2.

	Column 7	Column 16
2.	INSERT	Reference

This instruction operates as described in Paragraph C.2.1.

	Column 7	Column 16
3.	INCREMENT	Reference

This instruction causes the integer value of one to be added to the reference specified. If the reference is a variable, a single statement is generated adding one to the variable.

If the reference is a table reference without a maximum, the integer value of one is shifted left as necessary and added to the reference. If the reference is a table reference with a maximum value, the current contents are extracted, shifted as necessary, the value of one is added and compared to the maximum. After this test, the value is incremented by one. The reference should never be an integer constant.

	Column 7	Column 16
4.	DECREMENT	Reference

This instruction causes the integer value of one to be subtracted from the reference specified. If the reference is a variable, a single statement is generated subtracting one from the variable.

If the reference is a table reference, a test for a negative result is always made by extracting the current contents, and testing for a zero value. After the tests the reference is decremented by one.

	Column 7	Column 16
5.	CLEAR	Reference

If the reference specified is a variable, a single statement is generated setting the variable equal to integer 0.

If the reference is a table reference, the current contents of the reference are extracted and then subtracted from itself to cause a zero value for the item. The reference should never be an integer constant.

	Column 7	Column 16
6.	SET	Reference

The primary purpose of this instruction is to set a one-bit table reference to the value of one. For the IBM 360 computer the generated coding uses an OR function to set the right most bit without clearing the item. If the table reference is other than one bit in length, it is advisable to use the MOVE instruction in place of the SET. If the reference is a variable, a single statement is generated setting it equal to integer one. The reference should never be an integer constant.

	Column 7	Column 16	Column 26***	Column 31
7.	ADD	Reference	TO	Reference

The generated coding adds the contents of the first reference to the second, and the coding varies depending on whether the references are table references or variables. The first reference may be an integer constant. If the second reference is a table reference and has a maximum limit, a test is applied for overflow.

	Column 7	Column 16	Column 26***	Column 31
8.	SUBTRACT	Reference	FROM	Reference

As with the ADD instruction this coding varies depending on the references. If the second reference is a table reference which is not a full memory word, a test is made for a negative result. The first reference may be an integer constant.

	Column 7	Column 16	Column 26***	Column 31
9.	MOVE	Reference	TO	Reference

The generated coding varies according to the references. If the second reference is a table reference a test may be made for overflow. The first reference may be an integer constant.

	Column 7	Column 16	Column 26	Column 31	Columns 37-72
10.	TEST	Reference	xx	Reference	Imperative Statement

The TEST instruction extracts and shifts the contents of the references as necessary and generates a simple logical IF statement. Columns 26-27 must contain one of the six relational operators (EQ, NE, LT, LE, GE, or GT). The imperative statement appearing in Columns 37-72 of the pseudo-instruction card is transferred to the generated IF statement. Should the imperative statement require a continuation card, one or more may be used without a "P" in Column 1. The references may be integer constants. TO and FROM in Column 26 is optional to improve statement readability.

C.5 OPERATING INSTRUCTIONS FOR THE IBM 360

The Preprocessor program is currently operational on the IBM 360. The following control cards are required for program execution at the Department of Health, Education and Welfare Computer Center:

```
//STEPNAME EXEC PGM=CSCSCRPT,REGION=175K
//STEPLIB DD DSN=(Library, Unit, Vol, Disp)
//FT06F001 DD SYSOUT=A     (or other print device)
//FT12F001 DD DSN=&CSC,UNIT=SYSDA,SPACE=(TRK,(10,10)),
//                DISP=(NEW,PASS)
//FT10F001DD           (this data set contains reference tables, etc.)
//FT11F001 DD          (this data set contains program source deck)
//STEPNAME EXEC  FORCLGH,  PARM.COMP,='MAP,XL,SOURCE',REGION=224K
//COMP.SYSIN DD DSN=&CSC,DISP=(OLD,DELETE),UNIT=SYSDA
```

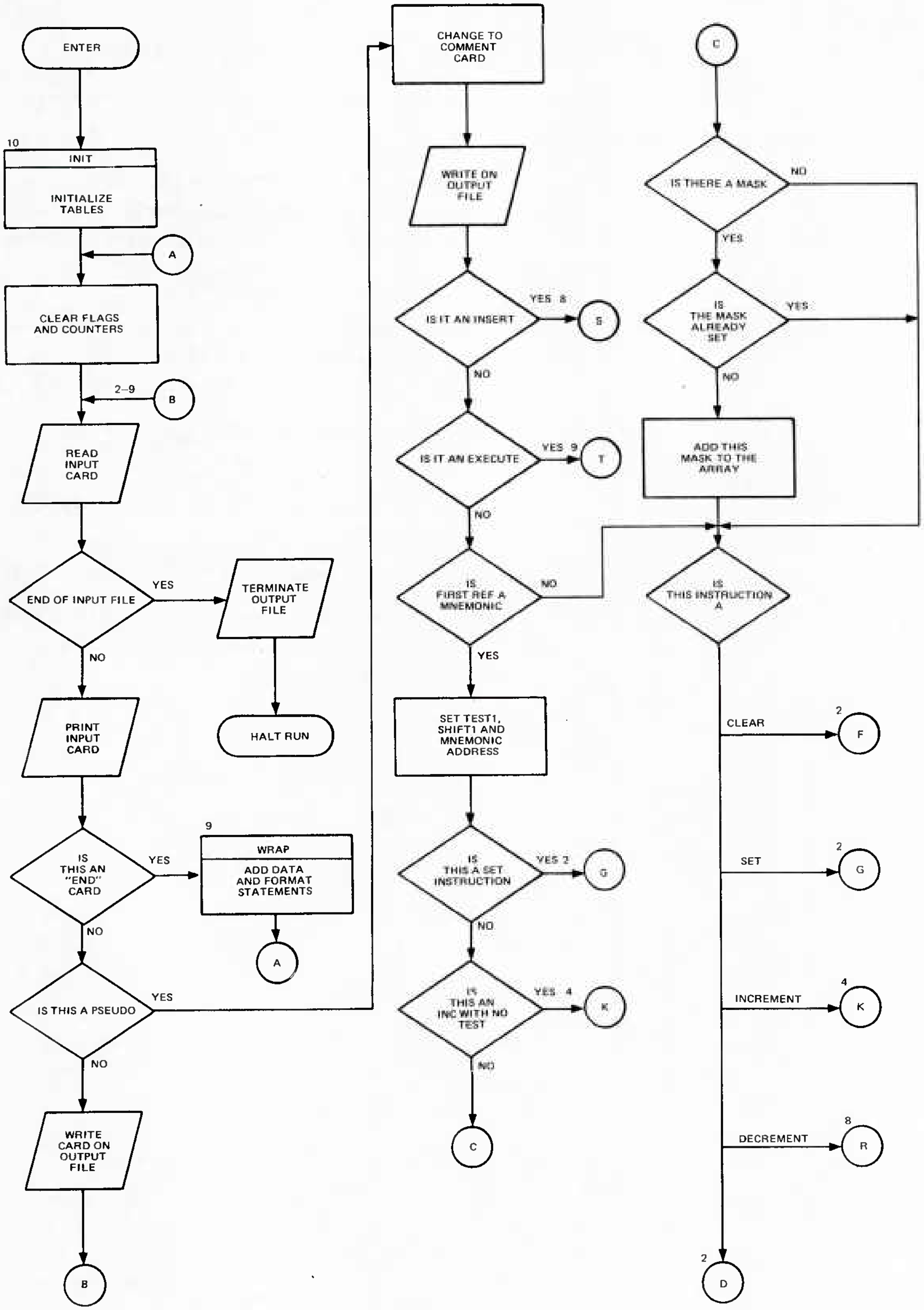
NOTE: To obtain bit manipulation, the H version FORTRAN Compiler must be used and XL must be included in the PARM.COMP parameters. This extended logic is not supported in the G Compiler.

Table C-1. Mnemonic Definition Card Format

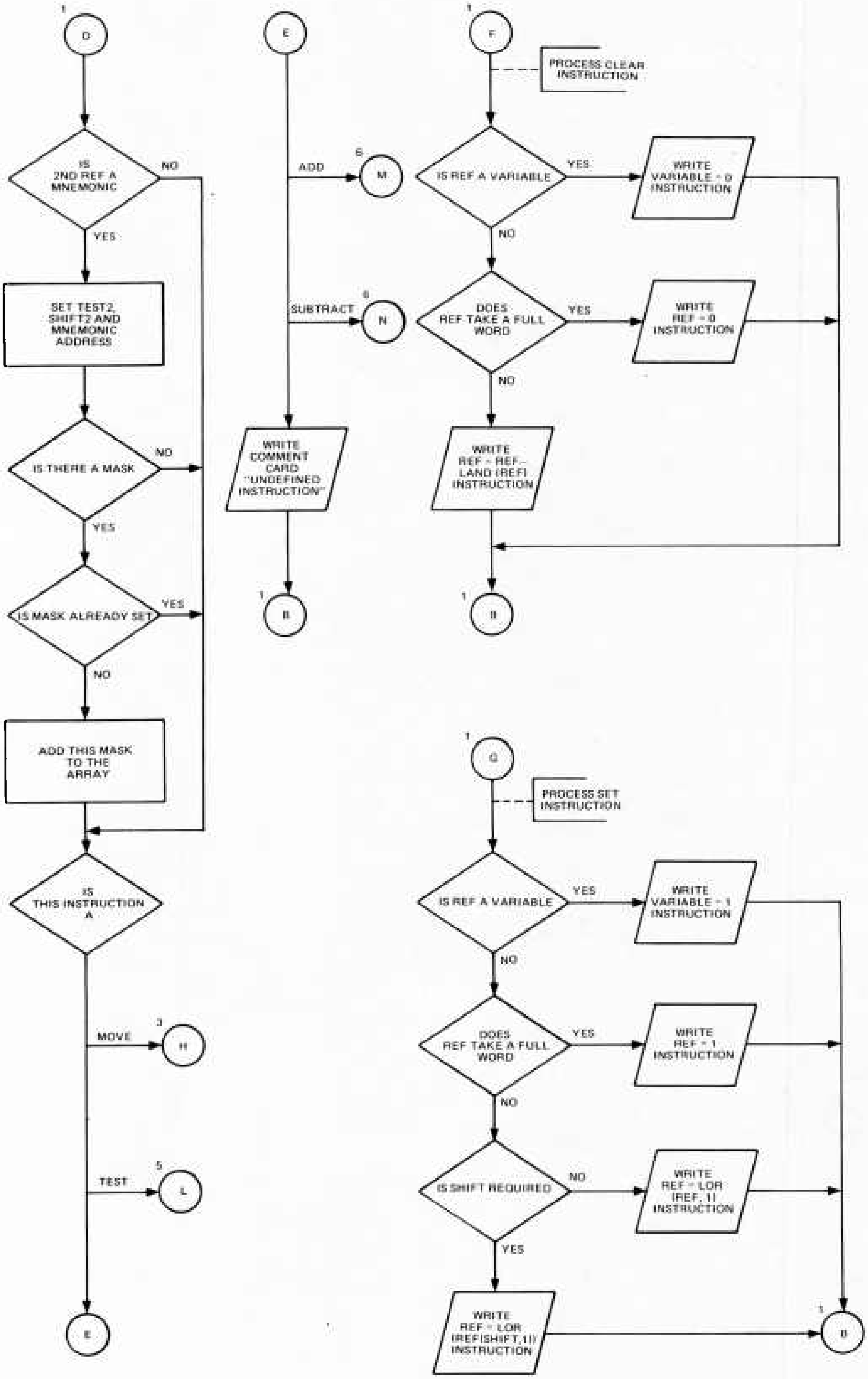
CARD COLUMN	DESCRIPTION	EXPLANATION
1-6	Reference Name	Mnemonic reference of one to six characters
7-8	Blank	
9-36	Reference Address	Address definition of the reference. Must be acceptable to compiler in terms of array names, subscripting, etc.
37	Blank	
38-45	Mask	Eight-hexidecimal character mask. If the reference occupies a full 4-byte word, this field should be blank.
46	Blank	
47-48	Shift	Integer value from 0 to 31 indicating amount of shift required to align field to normal integer position. Field may be blank. If a value is specified, a mask must also be used.
49	Blank	
50-59	Maximum value	Optional 10-digit entry which may specify a maximum value that a field may contain. Can be used to prevent one field from overflowing into an adjacent field.
60-80	Blank	

Table C-2. Pseudoinstruction Card Format

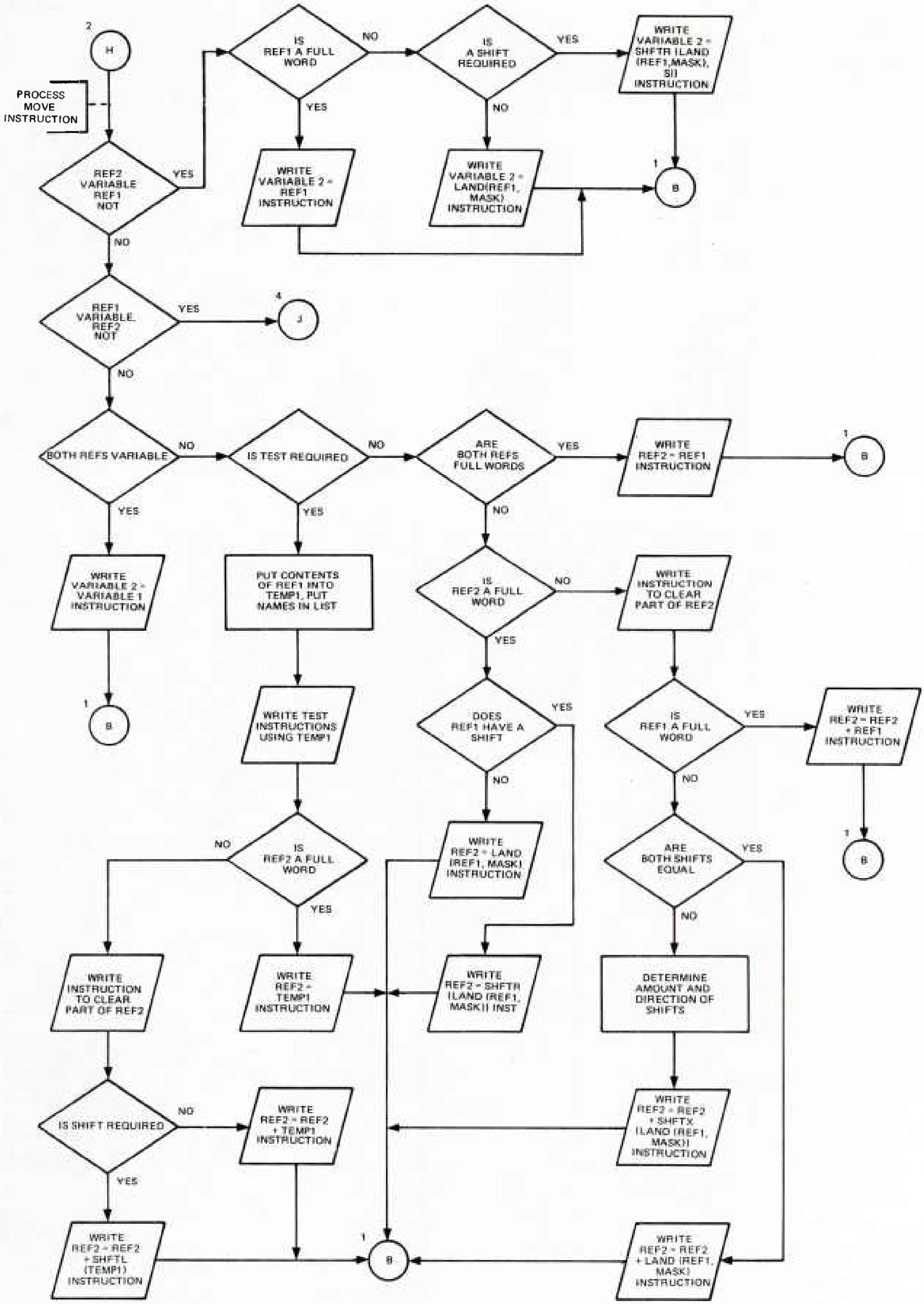
CARD COLUMN	DESCRIPTION	EXPLANATION
1	Pseudoinstruction flag	"P"
2-5	Statement Number	With exception of INSERT instruction, any statement number appearing in the pseudo-instruction will be transfered to first line of generated coding.
6	Blank	
7-15	Pseudoinstruction type	Preprocessor program only tests Columns 7-9. Instructions may be abbreviated to EXE, INS, INC, DEC, CLE, SET, ADD, SUB, MOV and TES.
16-21	First reference	First mnemonic reference or variable.
22-30	Optional entry	Immaterial except for the TEST instruction in which case Columns 26 and 27 must specify EQ, NE, LT, LE, GE, or GT.
31-36	Second reference	Second mnemonic reference or variable (if needed).
37-72	Optional entry	May be used for comments on all instructions except TEST, in which case a normally executable statement must be included.
73-80	Blank	



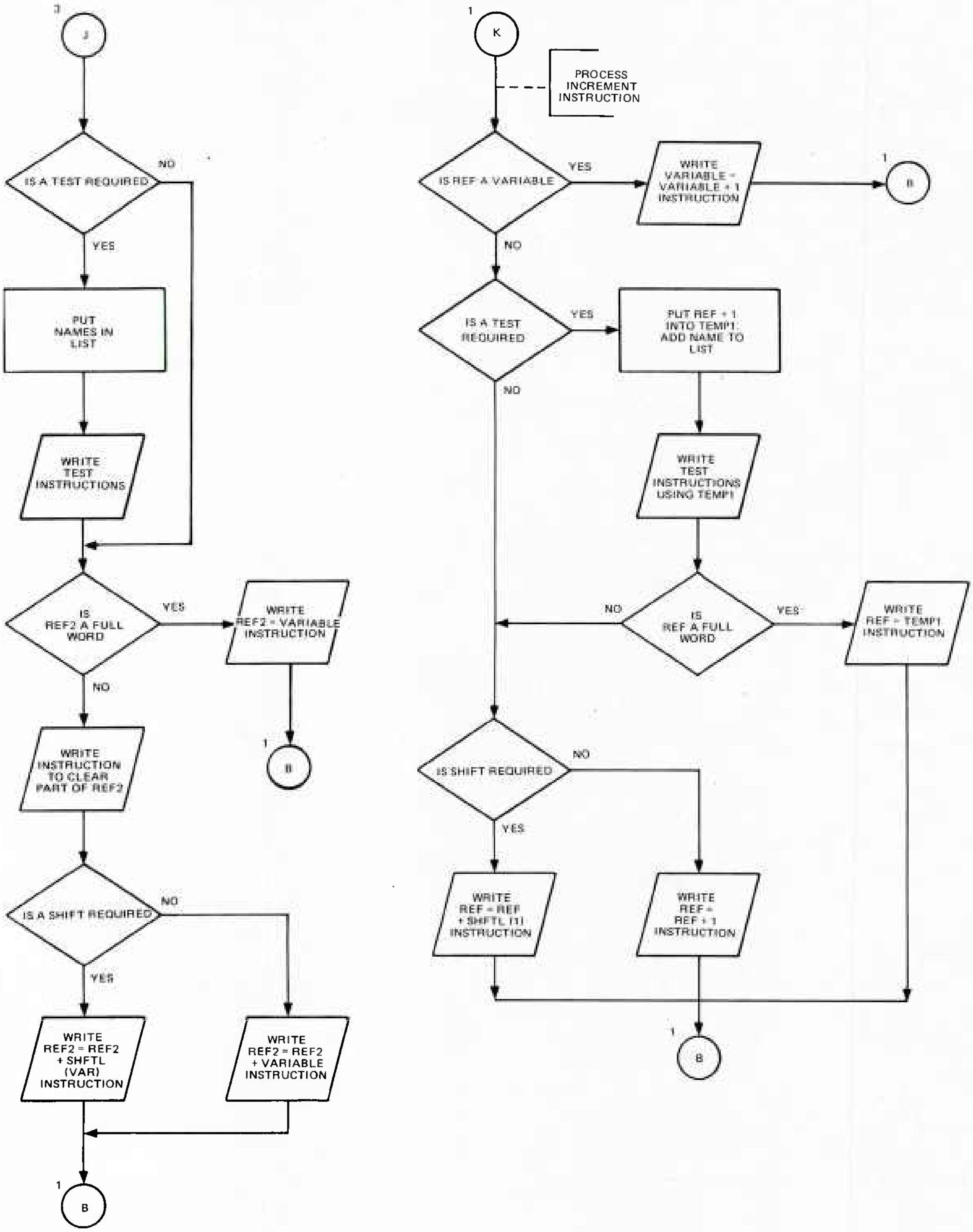
FORTRAN Preprocessor Program (Sheet 1 of 10)


1
D
IS 2ND REF A MNEMONIC
NO
YES
SET TEST2, SHIFT2 AND MNEMONIC ADDRESS
IS THERE A MASK
NO
YES
IS MASK ALREADY SET
YES
NO
ADD THIS MASK TO THE ARRAY
IS THIS INSTRUCTION A
MOVE
3
H
TEST
5
L
E
E
ADD
6
M
SUBTRACT
6
N
WRITE COMMENT CARD "UNDEFINED INSTRUCTION"
1
B
1
F
PROCESS CLEAR INSTRUCTION
IS REF A VARIABLE
YES
WRITE VARIABLE = 0 INSTRUCTION
NO
DOES REF TAKE A FULL WORD
YES
WRITE REF = 0 INSTRUCTION
NO
WRITE REF = REF – LAND (REF) INSTRUCTION
1
B
1
G
PROCESS SET INSTRUCTION
IS REF A VARIABLE
YES
WRITE VARIABLE = 1 INSTRUCTION
NO
DOES REF TAKE A FULL WORD
YES
WRITE REF = 1 INSTRUCTION
NO
IS SHIFT REQUIRED
NO
WRITE REF = LOR (REF, 1) INSTRUCTION
YES
WRITE REF = LOR (REFISHIFT,1)) INSTRUCTION
1
B

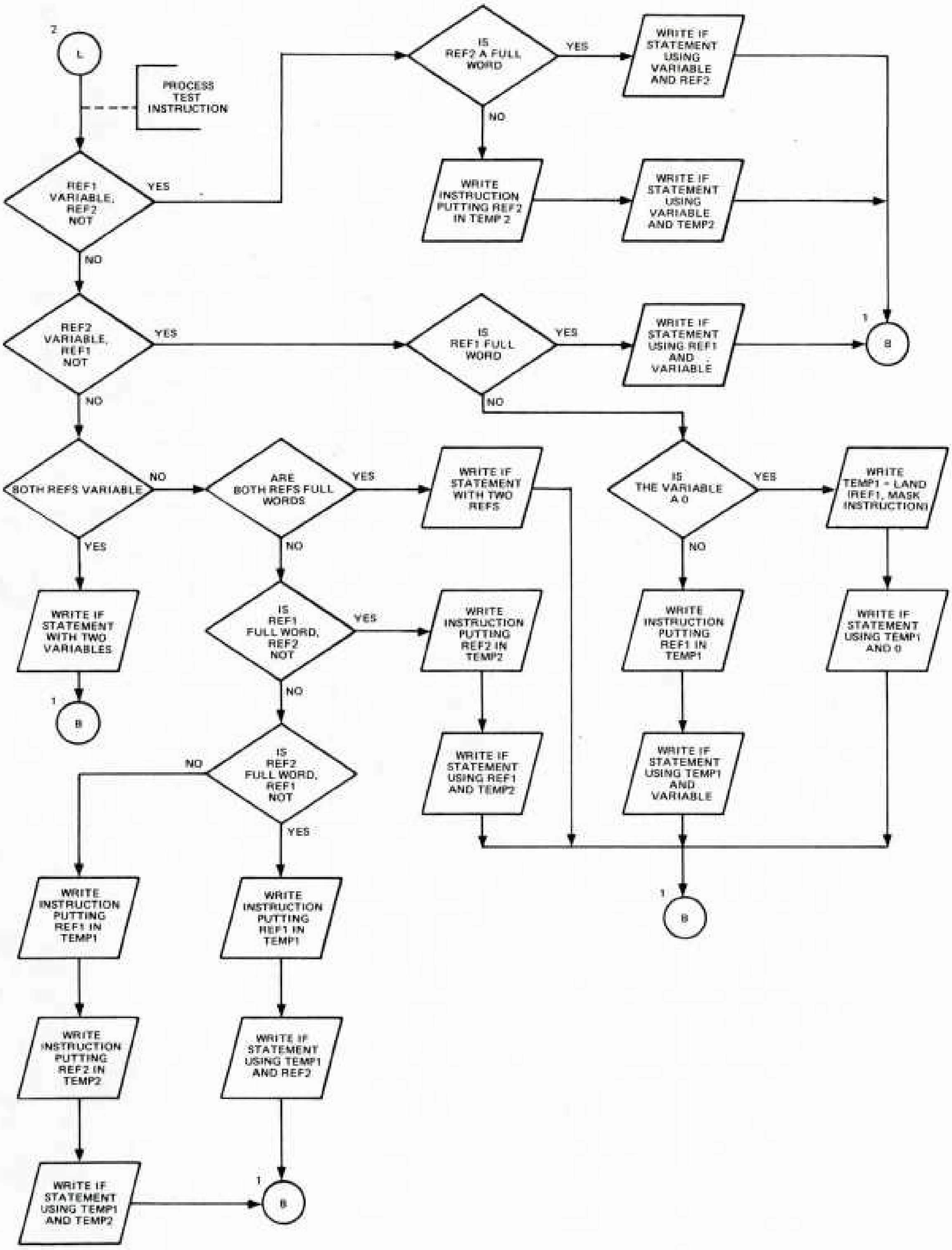
FORTRAN Preprocessor Program (Sheet 2 of 10)


2
H
PROCESS MOVE INSTRUCTION
REF2 VARIABLE REF1 NOT
YES
NO
IS REF1 A FULL WORD
NO
YES
IS A SHIFT REQUIRED
YES
NO
WRITE VARIABLE 2 = SHFTR (LAND (REF1,MASK), S)) INSTRUCTION
WRITE VARIABLE 2 = REF1 INSTRUCTION
WRITE VARIABLE 2 = LAND(REF1, MASK) INSTRUCTION
1
B
REF1 VARIABLE, REF2 NOT
YES
4
J
NO
BOTH REFS VARIABLE
NO
YES
IS TEST REQUIRED
NO
YES
ARE BOTH REFS FULL WORDS
YES
NO
WRITE REF2 = REF1 INSTRUCTION
1
B
WRITE VARIABLE 2 = VARIABLE 1 INSTRUCTION
1
B
PUT CONTENTS OF REF1 INTO TEMP1, PUT NAMES IN LIST
WRITE TEST INSTRUCTIONS USING TEMP1
IS REF2 A FULL WORD
NO
YES
WRITE INSTRUCTION TO CLEAR PART OF REF2
DOES REF1 HAVE A SHIFT
YES
NO
IS REF1 A FULL WORD
YES
NO
WRITE REF2 = REF2 + REF1 INSTRUCTION
1
B
IS REF2 A FULL WORD
NO
YES
WRITE REF2 = LAND (REF1, MASK) INSTRUCTION
ARE BOTH SHIFTS EQUAL
YES
NO
WRITE INSTRUCTION TO CLEAR PART OF REF2
WRITE REF2 = TEMP1 INSTRUCTION
WRITE REF2 = SHFTR (LAND (REF1, MASK)) INST
DETERMINE AMOUNT AND DIRECTION OF SHIFTS
IS SHIFT REQUIRED
NO
YES
WRITE REF2 = REF2 + TEMP1 INSTRUCTION
WRITE REF2 = REF2 + SHFTX (LAND (REF1, MASK)) INSTRUCTION
WRITE REF2 = REF2 + SHFTL (TEMP1) INSTRUCTION
1
B
WRITE REF2 = REF2 + LAND (REF1, MASK) INSTRUCTION

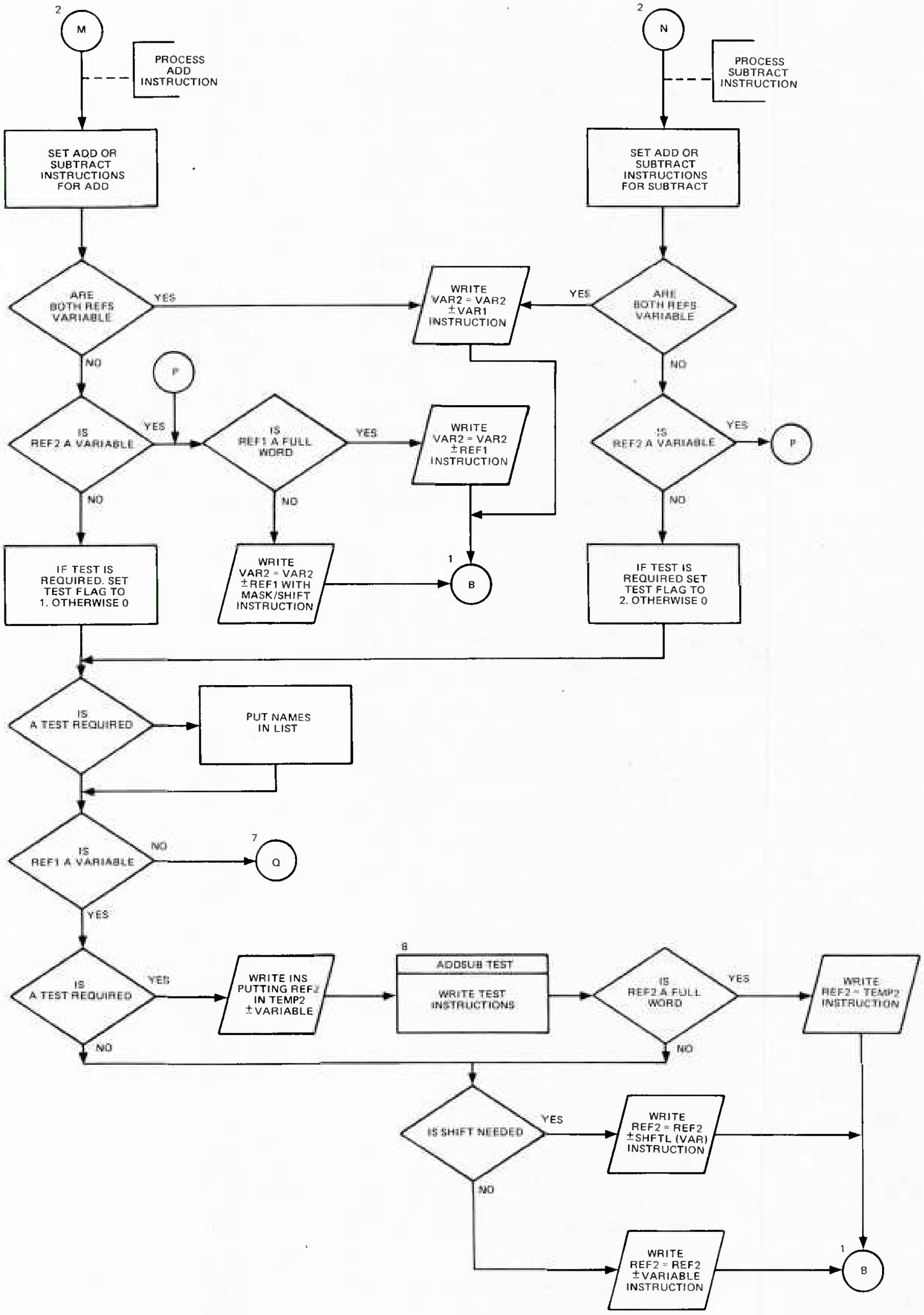
FORTRAN Preprocessor Program (Sheet 3 of 10)


3
J
IS A TEST REQUIRED
NO
YES
PUT NAMES IN LIST
WRITE TEST INSTRUCTIONS
IS REF2 A FULL WORD
YES
WRITE REF2 = VARIABLE INSTRUCTION
1
B
NO
WRITE INSTRUCTION TO CLEAR PART OF REF2
IS A SHIFT REQUIRED
NO
YES
WRITE REF2 = REF2 + SHFTL (VAR) INSTRUCTION
WRITE REF2 = REF2 + VARIABLE INSTRUCTION
1
B
1
K
PROCESS INCREMENT INSTRUCTION
IS REF A VARIABLE
YES
WRITE VARIABLE = VARIABLE + 1 INSTRUCTION
1
B
NO
IS A TEST REQUIRED
YES
PUT REF + 1 INTO TEMP1. ADD NAME TO LIST
NO
WRITE TEST INSTRUCTIONS USING TEMP1
IS REF A FULL WORD
NO
YES
WRITE REF = TEMP1 INSTRUCTION
IS SHIFT REQUIRED
NO
YES
WRITE REF = REF + SHFTL (1) INSTRUCTION
WRITE REF = REF + 1 INSTRUCTION
1
B

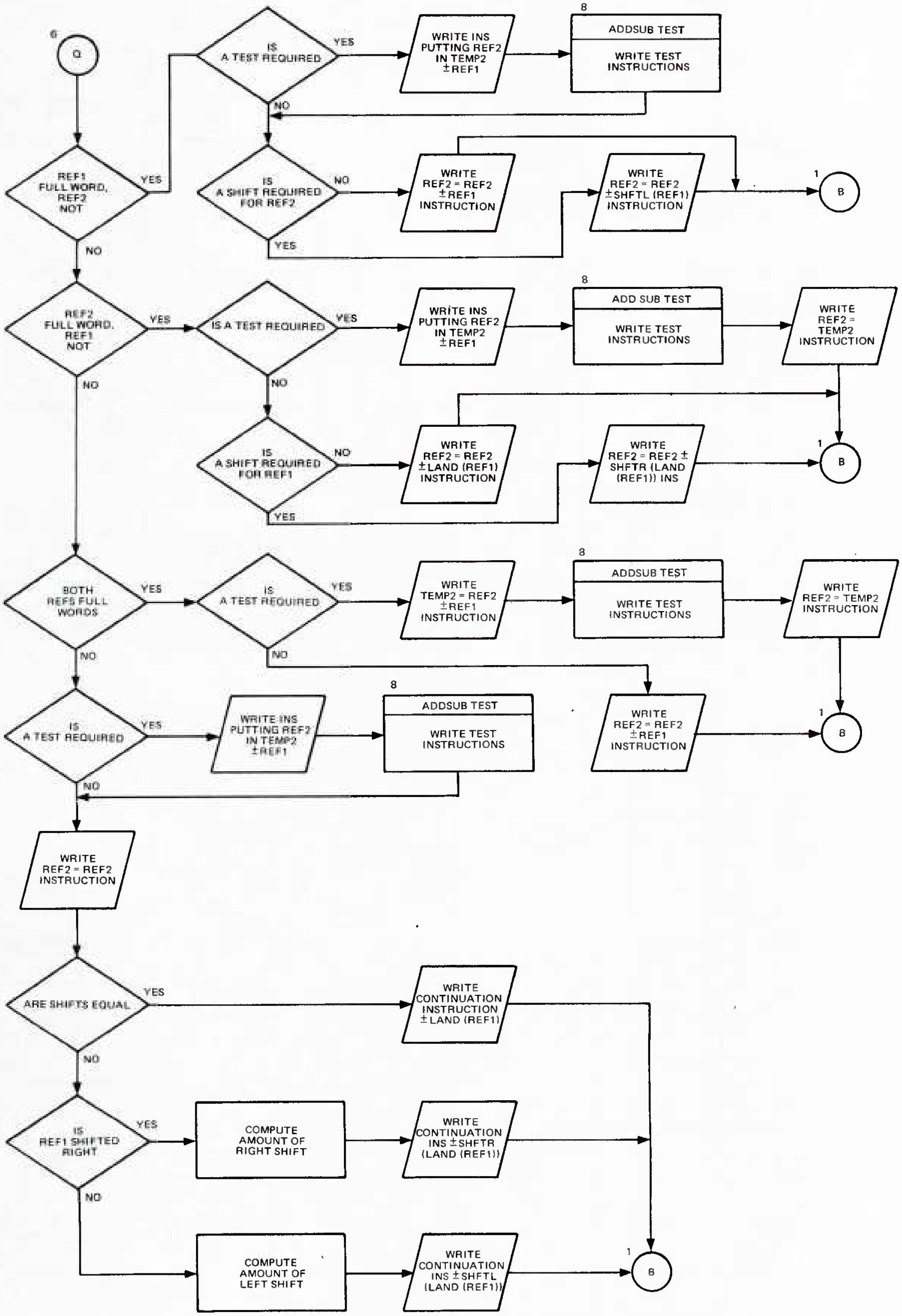
FORTRAN Preprocessor Program (Sheet 4 of 10)


2
L
PROCESS TEST INSTRUCTION
REF1 VARIABLE, REF2 NOT
YES
NO
IS REF2 A FULL WORD
YES
NO
WRITE IF STATEMENT USING VARIABLE AND REF2
WRITE INSTRUCTION PUTTING REF2 IN TEMP 2
WRITE IF STATEMENT USING VARIABLE AND TEMP2
REF2 VARIABLE, REF1 NOT
YES
NO
IS REF1 FULL WORD
YES
NO
WRITE IF STATEMENT USING REF1 AND VARIABLE
1
B
BOTH REFS VARIABLE
NO
YES
WRITE IF STATEMENT WITH TWO VARIABLES
1
B
ARE BOTH REFS FULL WORDS
YES
NO
WRITE IF STATEMENT WITH TWO REFS
IS REF1 FULL WORD, REF2 NOT
YES
NO
WRITE INSTRUCTION PUTTING REF2 IN TEMP2
WRITE IF STATEMENT USING REF1 AND TEMP2
IS REF2 FULL WORD, REF1 NOT
NO
YES
WRITE INSTRUCTION PUTTING REF1 IN TEMP1
WRITE INSTRUCTION PUTTING REF2 IN TEMP2
WRITE IF STATEMENT USING TEMP1 AND TEMP2
WRITE INSTRUCTION PUTTING REF1 IN TEMP1
WRITE IF STATEMENT USING TEMP1 AND REF2
1
B
IS THE VARIABLE A 0
YES
NO
WRITE TEMP1 = LAND (REF1, MASK INSTRUCTION)
WRITE IF STATEMENT USING TEMP1 AND 0
WRITE INSTRUCTION PUTTING REF1 IN TEMP1
WRITE IF STATEMENT USING TEMP1 AND VARIABLE
1
B

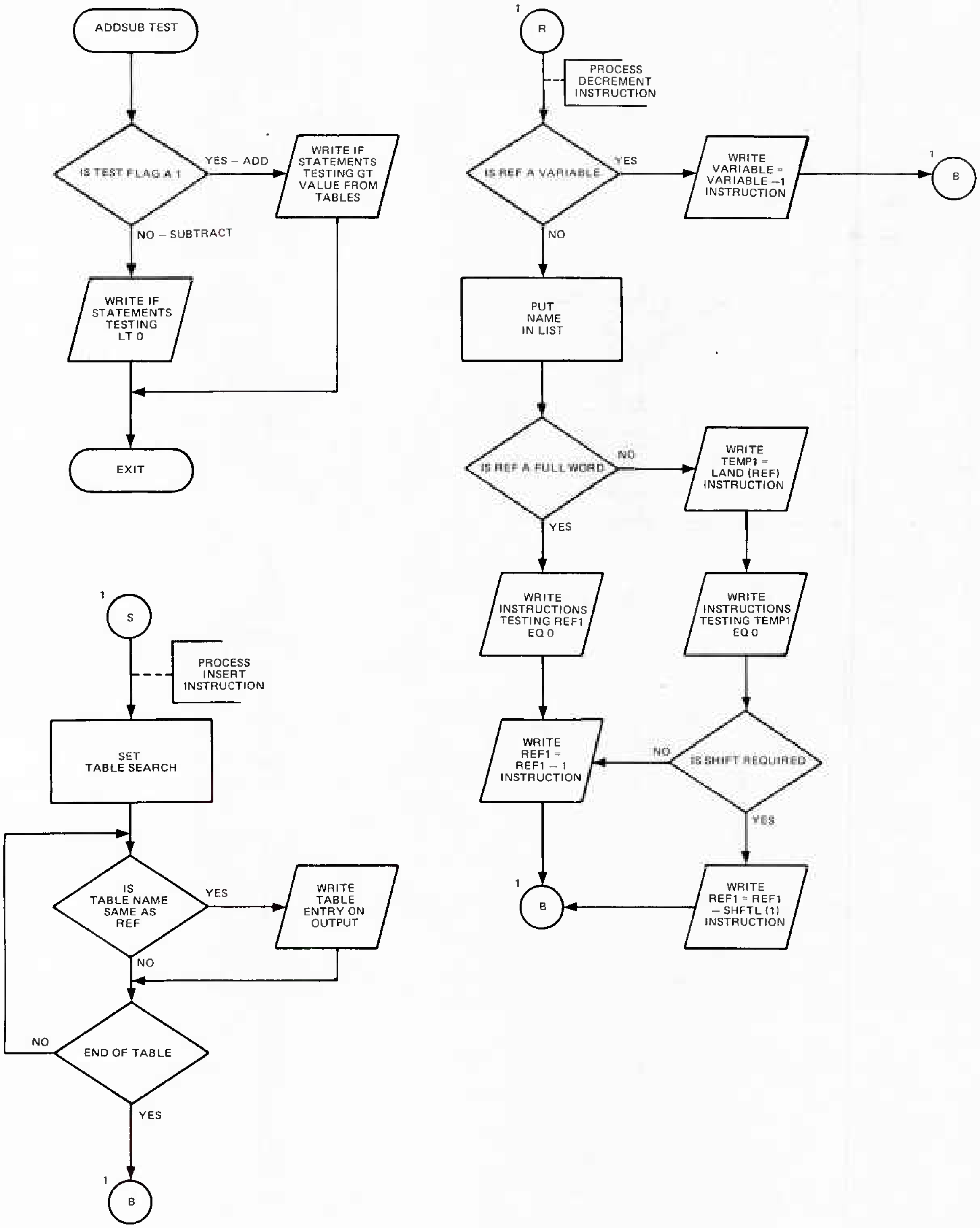
FORTRAN Preprocessor Program (Sheet 5 of 10)


2
M
PROCESS ADD INSTRUCTION
SET ADD OR SUBTRACT INSTRUCTIONS FOR ADD
ARE BOTH REFS VARIABLE
YES
NO
P
IS REF2 A VARIABLE
YES
NO
IS REF1 A FULL WORD
YES
NO
WRITE VAR2 = VAR2 ±VAR1 INSTRUCTION
WRITE VAR2 = VAR2 ±REF1 INSTRUCTION
IF TEST IS REQUIRED. SET TEST FLAG TO 1. OTHERWISE 0
WRITE VAR2 = VAR2 ±REF1 WITH MASK/SHIFT INSTRUCTION
1
B
2
N
PROCESS SUBTRACT INSTRUCTION
SET ADD OR SUBTRACT INSTRUCTIONS FOR SUBTRACT
YES
ARE BOTH REFS VARIABLE
NO
IS REF2 A VARIABLE
YES
P
NO
IF TEST IS REQUIRED SET TEST FLAG TO 2. OTHERWISE 0
IS A TEST REQUIRED
PUT NAMES IN LIST
IS REF1 A VARIABLE
NO
7
Q
YES
IS A TEST REQUIRED
YES
NO
WRITE INS PUTTING REF2 IN TEMP2 ±VARIABLE
8
ADDSUB TEST
WRITE TEST INSTRUCTIONS
IS REF2 A FULL WORD
YES
NO
WRITE REF2 = TEMP2 INSTRUCTION
IS SHIFT NEEDED
YES
NO
WRITE REF2 = REF2 ±SHFTL (VAR) INSTRUCTION
WRITE REF2 = REF2 ±VARIABLE INSTRUCTION
1
B

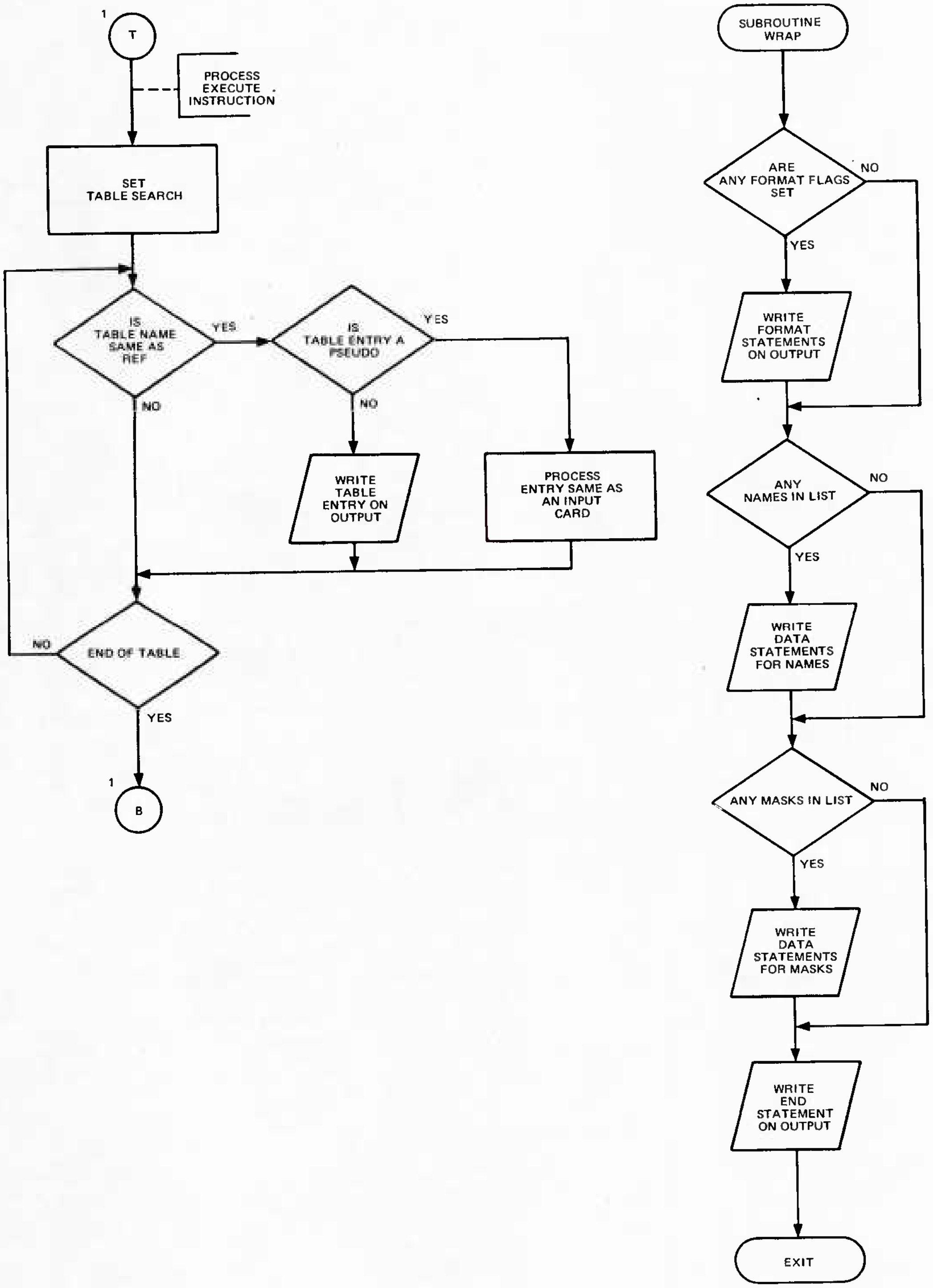
FORTRAN Preprocessor Program (Sheet 6 of 10)


6
Q
IS A TEST REQUIRED
YES
WRITE INS PUTTING REF2 IN TEMP2 ±REF1
8
ADDSUB TEST
WRITE TEST INSTRUCTIONS
NO
REF1 FULL WORD, REF2 NOT
YES
IS A SHIFT REQUIRED FOR REF2
NO
WRITE REF2 = REF2 ±REF1 INSTRUCTION
WRITE REF2 = REF2 ±SHFTL (REF1) INSTRUCTION
1
B
YES
NO
REF2 FULL WORD, REF1 NOT
YES
IS A TEST REQUIRED
YES
WRITE INS PUTTING REF2 IN TEMP2 ±REF1
8
ADD SUB TEST
WRITE TEST INSTRUCTIONS
WRITE REF2 = TEMP2 INSTRUCTION
NO
NO
IS A SHIFT REQUIRED FOR REF1
NO
WRITE REF2 = REF2 ±LAND (REF1) INSTRUCTION
WRITE REF2 = REF2 ± SHFTR (LAND (REF1)) INS
1
B
YES
BOTH REFS FULL WORDS
YES
IS A TEST REQUIRED
YES
WRITE TEMP2 = REF2 ±REF1 INSTRUCTION
8
ADDSUB TEST
WRITE TEST INSTRUCTIONS
WRITE REF2 = TEMP2 INSTRUCTION
NO
NO
IS A TEST REQUIRED
YES
WRITE INS PUTTING REF2 IN TEMP2 ±REF1
8
ADDSUB TEST
WRITE TEST INSTRUCTIONS
WRITE REF2 = REF2 ±REF1 INSTRUCTION
1
B
NO
WRITE REF2 = REF2 INSTRUCTION
ARE SHIFTS EQUAL
YES
WRITE CONTINUATION INSTRUCTION ±LAND (REF1)
NO
IS REF1 SHIFTED RIGHT
YES
COMPUTE AMOUNT OF RIGHT SHIFT
WRITE CONTINUATION INS ±SHFTR (LAND (REF1))
NO
COMPUTE AMOUNT OF LEFT SHIFT
WRITE CONTINUATION INS ±SHFTL (LAND (REF1))
1
B

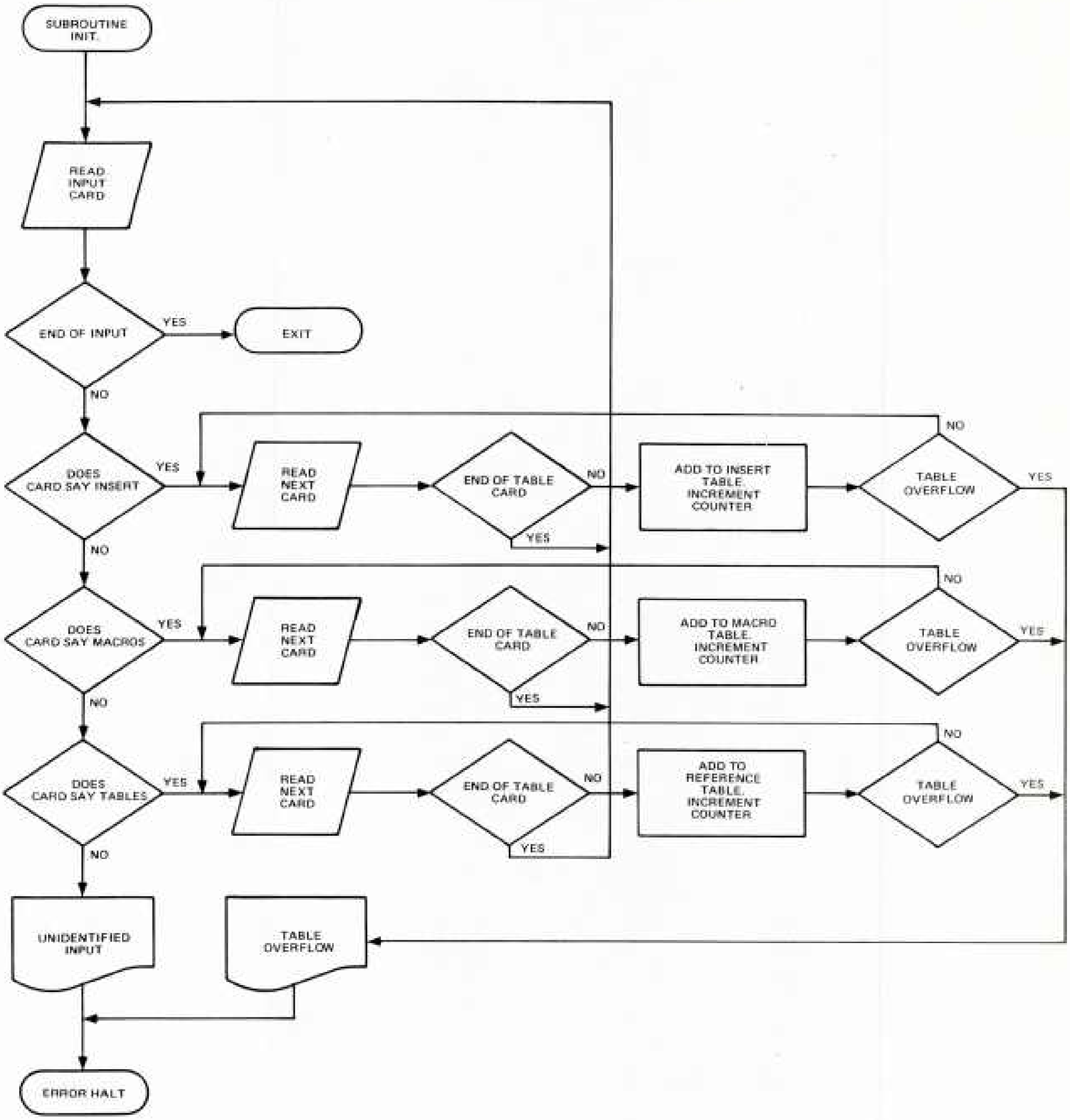
FORTRAN Preprocessor Program (Sheet 7 of 10)


ADDSUB TEST
IS TEST FLAG A 1
YES – ADD
WRITE IF STATEMENTS TESTING GT VALUE FROM TABLES
NO – SUBTRACT
WRITE IF STATEMENTS TESTING LT 0
EXIT
1
R
PROCESS DECREMENT INSTRUCTION
IS REF A VARIABLE
YES
WRITE VARIABLE = VARIABLE –1 INSTRUCTION
1
B
NO
PUT NAME IN LIST
IS REF A FULL WORD
NO
WRITE TEMP1 = LAND (REF) INSTRUCTION
YES
WRITE INSTRUCTIONS TESTING REF1 EQ 0
WRITE INSTRUCTIONS TESTING TEMP1 EQ 0
WRITE REF1 = REF1 – 1 INSTRUCTION
NO
IS SHIFT REQUIRED
YES
WRITE REF1 = REF1 – SHFTL (1) INSTRUCTION
1
B
1
S
PROCESS INSERT INSTRUCTION
SET TABLE SEARCH
IS TABLE NAME SAME AS REF
YES
WRITE TABLE ENTRY ON OUTPUT
NO
NO
END OF TABLE
YES
1
B

FORTRAN Preprocessor Program (Sheet 8 of 10)


1
T
PROCESS EXECUTE INSTRUCTION
SET TABLE SEARCH
IS TABLE NAME SAME AS REF
YES
NO
IS TABLE ENTRY A PSEUDO
YES
NO
WRITE TABLE ENTRY ON OUTPUT
PROCESS ENTRY SAME AS AN INPUT CARD
END OF TABLE
NO
YES
1
B
SUBROUTINE WRAP
ARE ANY FORMAT FLAGS SET
NO
YES
WRITE FORMAT STATEMENTS ON OUTPUT
ANY NAMES IN LIST
NO
YES
WRITE DATA STATEMENTS FOR NAMES
ANY MASKS IN LIST
NO
YES
WRITE DATA STATEMENTS FOR MASKS
WRITE END STATEMENT ON OUTPUT
EXIT

FORTRAN Preprocessor Program (Sheet 9 of 10)



FORTRAN Preprocessor Program (Sheet 10 Of 10)

APPENDIX D - TABLE STRUCTURE FOR IBM 360 SYSTEM

Table D-1. Node Status Table Structure

<table>
<tr><th>WORD</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>1</td><td colspan="16">MPICS</td><td colspan="16">MPICE</td></tr>
<tr><td>2</td><td colspan="16">MPOQS</td><td colspan="16">MPOQE</td></tr>
<tr><td>3</td><td colspan="16">MPIQS</td><td colspan="16">MPIQE</td></tr>
<tr><td>4</td><td colspan="16">NSFL</td><td colspan="16">NSRT</td></tr>
<tr><td>5</td><td colspan="16">NSFT</td><td colspan="16">NSQIT</td></tr>
<tr><td>6</td><td colspan="16">MPCVNT</td><td colspan="16">NSQOT</td></tr>
<tr><td>7</td><td colspan="8">NSNT</td><td colspan="8">NSNL</td><td colspan="16">NSQL</td></tr>
<tr><td>8</td><td colspan="16">MPTQC</td><td colspan="16">MPORL</td></tr>
<tr><td>9</td><td colspan="8"></td><td colspan="3">NSNLT</td><td colspan="20">MPLTCD</td><td>NSNO</td></tr>
<tr><td>10</td><td colspan="11"></td><td colspan="20">MPLTCX</td><td>MPNLTC</td></tr>
</table>

Table D-1. Node Status Table Structure (Continued)

<table>
<tr><th>WORD</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>11</td><td colspan="11"></td><td colspan="20">MPOFG</td><td>MPNFG</td></tr>
<tr><td>12</td><td colspan="9"></td><td colspan="20">FETT</td><td colspan="3">MPPROG</td></tr>
<tr><td>13</td><td colspan="11"></td><td colspan="21">FETU</td></tr>
<tr><td>14</td><td colspan="32">MPIPID</td></tr>
<tr><td>15</td><td colspan="32">MPISOM</td></tr>
<tr><td>16</td><td colspan="32">MPOPOD</td></tr>
<tr><td>17</td><td colspan="32">MPOSOM</td></tr>
<tr><td>18</td><td colspan="32">MPIRIL</td></tr>
<tr><td>19</td><td colspan="32">MPIRIH</td></tr>
<tr><td>20</td><td colspan="32">MPOMX</td></tr>
</table>

Table D-1. Node Status Table Structure (Continued)

<table>
<tr><th>WORD</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>21</td><td colspan="32">MPMISC</td></tr>
<tr><td>22</td><td colspan="11"></td><td colspan="20">FELBA1</td><td>FETI1</td></tr>
<tr><td>23</td><td colspan="11"></td><td colspan="20">FELBA2</td><td>FETI2</td></tr>
<tr><td>24</td><td colspan="11"></td><td colspan="20">FELBA3</td><td>FETI3</td></tr>
<tr><td>25</td><td colspan="11"></td><td colspan="20">FELBA4</td><td>FETI4</td></tr>
<tr><td>26</td><td colspan="16">MPTQS</td><td colspan="16">MPTQE</td></tr>
</table>

Table D-2. Link Status Table and Link Channel Status Table

<table>
<tr><th>WORD</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th><th></th></tr>
<tr><td>1</td><td colspan="2"></td><td colspan="16">LTQS</td><td colspan="8">LTCA</td><td colspan="5">LTSN</td><td>LTTN</td><td rowspan="5">LINK</td></tr>
<tr><td>2</td><td></td><td colspan="16">LTQE</td><td colspan="8">LTCB</td><td>LTOIP</td><td>LT</td><td>LTC</td><td colspan="3">LTSC</td><td>LTLO</td></tr>
<tr><td>3</td><td colspan="7"></td><td colspan="16">LTQP</td><td colspan="8">LTNC</td><td>LTQF</td></tr>
<tr><td>4</td><td colspan="8"></td><td colspan="6">LNDN</td><td>LNAA</td><td colspan="16">LNQHI</td><td>LTPRE</td></tr>
<tr><td>5</td><td colspan="16">LNDL</td><td colspan="16">LNQLO</td></tr>
<tr><td>6</td><td colspan="10"></td><td colspan="16">CHEV</td><td>CHPRA</td><td>CHPR</td><td>CHDFLG</td><td>CHEM</td><td>CHSM</td><td>CHCO</td><td rowspan="2">CHANNEL 1</td></tr>
<tr><td>7</td><td colspan="2"></td><td colspan="15">LCAD</td><td colspan="15">LCSP</td></tr>
<tr><td>8</td><td colspan="10"></td><td colspan="16">CHEV</td><td>CHPRA</td><td>CHPR</td><td>CHDFLG</td><td>CHEM</td><td>CHSM</td><td>CHCO</td><td rowspan="2">CHANNEL 2</td></tr>
<tr><td>9</td><td colspan="2"></td><td colspan="15">LCAD</td><td colspan="15">LCSP</td></tr>
<tr><td>10</td><td colspan="10"></td><td colspan="16">CHEV</td><td>CHPRA</td><td>CHPR</td><td>CHDFLG</td><td>CHEM</td><td>CHSM</td><td>CHCO</td><td rowspan="2">CHANNEL n</td></tr>
<tr><td>11</td><td colspan="2"></td><td colspan="15">LCAD</td><td colspan="15">LCSP</td></tr>
</table>

Table D-3. Tributary Status Table and Tributary Channel Status Table

<table>
<tr><th>WORD</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th><th></th></tr>
<tr><td>1</td><td colspan="2"></td><td colspan="16">LTQS</td><td colspan="8">LTCA</td><td colspan="5">LTSN</td><td>LTTN</td><td rowspan="7">TRIBUTARY</td></tr>
<tr><td>2</td><td></td><td colspan="16">LTQE</td><td colspan="8">LTCB</td><td>LTOIP</td><td colspan="2">LTLTC</td><td colspan="3">LTSC</td><td>LTLO</td></tr>
<tr><td>3</td><td colspan="7"></td><td colspan="16">LTQP</td><td colspan="8">LTNC</td><td>LTQF</td></tr>
<tr><td>4</td><td colspan="15">TRSPT</td><td colspan="16">TRIQ</td><td>LTPPE</td></tr>
<tr><td>5</td><td colspan="15">TRSPM</td><td colspan="16">TROQ</td><td>TRQFI</td></tr>
<tr><td>6</td><td></td><td colspan="15">TRSPC</td><td colspan="16">TRINQ</td></tr>
<tr><td>7</td><td></td><td colspan="15">TRAD</td><td colspan="16">TRVN</td></tr>
<tr><td>8</td><td colspan="10"></td><td colspan="16">CHEV</td><td>CHPPA</td><td>CHPR</td><td>CHDFLG</td><td>CHEM</td><td>CHSM</td><td>CHCO</td><td>INPUT CHANNEL I</td></tr>
<tr><td>9</td><td colspan="10"></td><td colspan="16">CHEV</td><td>CHPPA</td><td>CHPR</td><td>CHDFLG</td><td>CHEM</td><td>CHSM</td><td>CHCO</td><td>OUTPUT CHANNEL I</td></tr>
<tr><td>10</td><td colspan="10"></td><td colspan="16">CHEV</td><td>CHPPA</td><td>CHPR</td><td>CHDFLG</td><td>CHEM</td><td>CHSM</td><td>CHCO</td><td>INPUT CHANNEL N</td></tr>
<tr><td>11</td><td colspan="10"></td><td colspan="16">CHEV</td><td>CHPPA</td><td>CHPR</td><td>CHDFLG</td><td>CHEM</td><td>CHSM</td><td>CHCO</td><td>OUTPUT CHANNEL N</td></tr>
</table>

Table D-4. Message Status Table

<table>
<tr><th>WORD</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>1</td><td colspan="16">MSDN2</td><td colspan="8">MSRI</td><td colspan="4">MSPR</td><td colspan="2">MSTP</td><td>M
S
A
L
T</td><td>M
S
X
F</td></tr>
<tr><td>2</td><td colspan="20">MSNUM</td><td colspan="9">MSLN</td><td colspan="3">MSSC</td></tr>
<tr><td>3</td><td colspan="16">MSDN1</td><td colspan="16">MSLR</td></tr>
</table>

Table D-5. Message Continuation Status Table

<table>
<tr><th>WORD</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>1</td><td colspan="16">MCDN4</td><td colspan="16">MCDN1</td></tr>
<tr><td>2</td><td colspan="16">MCDN5</td><td colspan="16">MCDN2</td></tr>
<tr><td>3</td><td colspan="16">MCDN6</td><td colspan="16">MCDN3</td></tr>
</table>

Table D-6. Transmission Event Status Table

WORD	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1	EVLSA																EVBL															
2	EVNS					EVT																										
3	EVMS																EVCS								EVTP							

Table D-7. Linkage Status Table

WORD	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1	EVLK																EVBL															